AL-RYADA JOURNAL FOR COMPUTATIONAL INTELLIGENCE AND TECHNOLOGY (ARJCIT)

VOL. 1 (1) - 2024

Detection Techniques for Two Critical Nodes in Ad Hoc Networks

Mokhtar A. A. Mohamed [a,b],

[a] *Comp. Sci. & Eng. Dept., Faculty of Electronic Engineering, Menoufia University, Egypt.*
[b] *Faculty of Computers & AI , ALRYADA University for Science and Technology, Egypt.*

Article history:

Received: 22 – Aug – 2024

Revised: 22 – Aug – 2024

Accepted: 05 – Oct – 2024

Available online: 1 - DEC - 2024

DOI:

10.21608/ajcit.2024.302421.1004

Correspondence

Mokhtar A. A. Mohamed,

Faculty of Computers & AI (FCAI), ALRYADA University for Science & Technology (RST), Egypt.

Email:
Mokhtar.Mohamed@rst.edu.eg

ABSTRACT:

Recently, one of the important requirements, in ad hoc networks, is to avoid partitioning problem due to dynamic movement of special node(s) called "Critical Node(s)". Critical node refers to the node that its removal from the network will break it into many partitions. Many researches presented different algorithms to detect such critical node(s) in any ad hoc network. These researches did not take the situation if two nodes together can cause also network separation if both of them are moved or removed at the same time. This paper introduces a new technique called "Two Critical-Nodes Detection Algorithm (TCNDA)" to determine individual critical node(s) and also to determine each two-node pair(s) which are critical together.

KEYWORDS

Ad hoc networks ; critical nodes ; localized algorithms ;

1. INTRODUCTION

The area of Ad hoc networks is very active nowadays due to its infrastructure-less nature, simplicity of deployment in areas where infrastructure hard to established, mobility and self-configurable nature make it suitable for huge range of applications. These applications include military and battlefield scenarios and disaster situations where no infrastructure is present, search and rescue jobs, transmission of road and weather conditions, taxicab network and inter-vehicle networks and educations operation by establish virtual classrooms and ad hoc communications through meetings or lectures [1-3].

Ad hoc network is a group of wireless mobile nodes creating a temporary network without any fixed infrastructure and no need to centralized administration. Nodes in ad hoc network communicate with each other in a peer-to-peer style where any two nodes are near enough to each other are assumed to have a direct radio link.

The maximum distance enough to establish the direct radio link to other node is called the node's transmission range. Each node plays two important roles: host and router to forward the packets of other nodes .To accomplish a connected ad hoc network; there must be a direct path of multi-hop path from a node to every other node [4].

In any network, critical node represents the node which its removal from the network will break the network into many partitions. As shown in Figure.1, both node (1) and (2) are critical nodes. Nodes inside each partition are capable of communicate with each other but nodes in other partitions are inaccessible. There are many applications of detection of critical node. The applications involve situations where the aim is to either protect the connectivity of nodes in a network by securing the most critical nodes or attacking the most critical nodes in order to have minimum connections between all pairs of nodes in the network [5]. Some of ad hoc network applications are very critical from being separated such as communication between members of a rescue team or between soldiers in a battlefield. In such situations, human life may be depending on the availability of this communication network. So, the detection of critical node is very important as first step to take care of it to avoid its failure or removal.

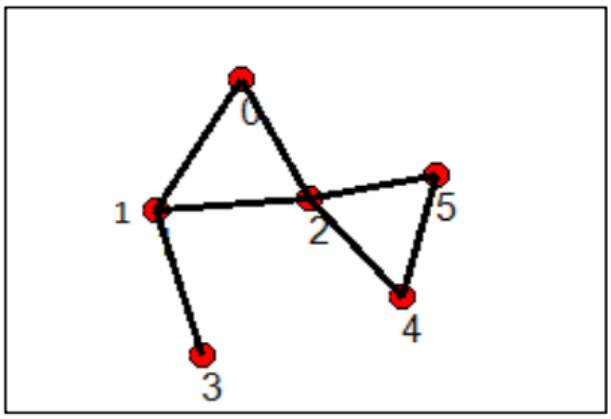

Figure 1: An Example of a small network

This paper provides a literature review of the most recent methods that concern with the detection of critical node in ad hoc networks. Then, it presents novel algorithm for detection of critical nodes in Ad-hoc networks. The proposed approach is called Two Critical Nodes Detection Algorithm (TCNDA). This approach enables each node to determine if it's local critical or there is another node from its k-hop neighbors and both of them together are critical. Finally, the proposed approach is evaluated and the effectiveness of TCNDA in determining the criticality of the node is validated through simulation experiments. Simulation results demonstrate that the performance of TCNDA with local k-hop information is quite competitive compared to centralized algorithm with global network information.

The rest of this paper is organized as follows. Section 2 introduces some of related work. Section 3 presents the TCNDA algorithm while Section 4 describes the performance evaluation of proposed algorithms. Finally, concluding remarks are listed in Section 5.

2. RELATED WORK

The algorithms concern with the detection of critical node can be classified into centralized and distributed based on the way of execution. Centralized algorithms [6, 7] required a node to have global information about the all-network topology. In [6], Duque-Anton et al. used depth first search (DFS) In order to detect the critical nodes, the rest connectivity value was introduced, which defined as the number of node pairs which got disconnected when the separation element is removed from the graph. In [7] they extend the depth first search (DFS) used in [6] to detect the links what they name as critical links. Critical links can be defined in several ways. One possible definition is that a link AB is critical if both nodes A and B are critical nodes. But two critical nodes may have another alternate path between them. So, a better way to define critical link is that the link connecting two critical nodes so that, when this link is removed from the graph, the graph becomes disconnected. However Centralized algorithms required keeping global information about topology which involves very communication overhead and thus may not be suitable in ad hoc network.

Oppositely, distributed algorithms don't need network keeping global information at any node and can be used for ad hoc network. For example, DDFS (Distributed Depth-First Search) which has been discussed in [8] implements DFS in the graph in a distributed manner. But they show by experiments that the time delay of DDFS is much higher, and it is more sensitive than the other algorithms to failures of links and nodes. Xiaomei et al. [9] presented a distributed mechanism to detect cut vertices in overlay networks called CAM (Connection Adjacency Matrix). In CAM, every cut candidate sends a component probe message to each neighbor, which contains timestamp, a TTL threshold indicating the expiration time of the message. Every node when receives a new message, it updates its local knowledge and make a choice to forward the message or to return arrival message. Each candidate updates its CAM graph based on the returned arrival messages. The candidate node is a cut vertex if its CAM graph has more than one component. However, experiments and results in [8] shows that CAM takes too large communication cost, which is not suitable for ad hoc networks. Xiong et al. [8] presented a distributed algorithm called CVD (Cut Vertex Detection) to detect cut vertices. CVD travels through the nodes of a network in parallel and colors the edges based on the interval-coded spanning tree. Then it decides the cut vertices by counting the edge colors. Sheng et al. [10] presented a distributed algorithm called DMCC (Detection algorithm based on Midpoint Coverage Circle). DMCC constructs an area (Midpoint Coverage Circle) with node n as the center to detect whether node n is a critical node in the subgraph G*. if node i is detected as a critical node in G*, it should find all possible global paths of node i neighbors by using RREQ (Route Request) and RREP (Route Reply) to make a decision if it is a global critical or not. If the node is non-critical in G* there is no need to check in the global. So, this will reduce communication overhead and will increases speed.

Both Jorgic et al. [11] and Imran et al. [12] show how to find a critical node in k-hop information and present the accuracy of k-hop critical nodes by comparing it with the global critical node. These algorithms only determine one critical node when removed the network is separated into groups and don't take the situation if two nodes together are removed the network separated. However, our proposed TCNDA algorithm determines the critical nodes and two nodes together as critical in K-hop information by making updating of LASCNN algorithm descried in [12].

3. PROPOSED ALGORITHM

The TCNDA determine the critical nodes and two nodes together as critical using local information. Local information is dependent on k-hop notation, two nodes are k-hop neighbors if the shortest path between them has k or less hops. If node has information about itself only, we say it has 0-hop information, but if node has information about direct neighbors only then it has 1-hop information, also if it has information about the neighbors of its direct neighbors then it has 2-hop information and so on.

3.1 Builds k-hop connection list

Nodes collect their local information of k-hop neighbors after discovering each other by sending "hello" messages to its neighbors containing the information of their (k-1)-hop neighbors. The1-hop information is a list of direct neighbors with their geographic positions. The 2-hop information is gotten by spreading lists of 1-hop neighbors and so on. Using the k-hop information nodes establish a k-hop connection list. For example, a 1-hop connection list of nodes (1) for the network Figure.1 is shown in Table 1 where each row is a connection between two nodes.

Table 1: 1-hop connection list of node (1).

1-hop connection list	
1 <> 0	direct neighbours
1 <> 2	direct neighbours
1 <> 3	direct neighbours
0 <> 2	determined by positions

A 1-hop connection list holds the direct neighbors of a node in the form of a pair. the connection between the neighbors is determine by calculation of distance between them using their geographic positions and compare it with the radio range of the node, usually this radio range are constant for all nodes. For example, node (1) can determine the connection between its 1- hop neighbors Node (0) and node (2) based on their position. Similarly, by collection the 1-hop connection list of the 1-hop neighbors without duplication, 2-hop connection list can be established. Also, the established 2-hop connection list will contain the connections between 1-hop neighbors and between a1-hop and 2-hop neighbors, but connection between 2-hop neighbors can be determined based on the distance between the nodes like in constructing the 1-hop connection list. If we will have 2-hop connection list there is no need to determine the connection between 1-hop neighbors when build the 1-hop connection list because it will be involved in 2-hop connection list .In the same way if we work in k-hop connection list it will have the connection between (k-1)-hop and k hop and we need to determine the connection between k-hop neighbors only and no need to determine it in any other steps. Table2 shows a 2-hop connection list of nodes (1).

Keeping the connection list updated is very important because topology of ad hoc network continuously changes due to network dynamics. Nodes in ad hoc networks always exchange update messages to display any changes in topology and share the criticality of nodes. Once a node receiving an update message it forwards the message to (k-1)-hop neighbours.so every node will update its connection list based on these massages.

Table 2: 2-hop connection list of node (1).

2-hop connection list
1 < > 0
1 < > 2
1 < > 3
0 < > 2
2 < > 5
2 < > 4
4 < > 5

3.2 Detection phase

After each node builds its k-hop connection list it will process it to decide whether it is critical node or not. If it is not critical, it will decide whether there is another node connected with it and both of them together are critical.

First step in detection phase is check the number of connection in the connection list, if the number of connection is one connection only, then the node is leaf node and the remove of it from the network will not disconnect other node, so it will be not critical and there is no chance to be critical together with other node for example node (3) in Figure 1 is leaf node. On the other hand, if the number of connections is more than one, the node creates a list of neighbors' nodes which are connected without it called connected-neighbors list and it will be empty in the first iteration. After that the node start processing its connection list one by one, if the connection has the node tested itself then ignore the connection and go to the next connection. If the connected- neighbors list is empty, then add the two nodes of the connection into the connected-neighbors list. Otherwise, if the connection has a common node with the connected neighbors the other node will be added into the connected neighbors. After processing all connection in the connection list, we check the size of the connected-neighbors list, if the size is increase from the last iteration, then repeat the iteration. In this manner, the connected-neighbors list will have all the connected neighbors without the node. If the size of the connected-neighbors list is equal to the number of k-hop neighbors of the node, node is not critical. But if the size is less then node is critical.

For example, if we test node (9) in figure 2, TCNDA algorithm needs four iterations to build node (9) connected-neighbors list as shown in table 3 (column 3). At the beginning connected neighbors list is empty. At first iteration, TCNDA will ignore connections (9 < > 6) and (9 < > 0) because these connections have node (9) itself. Then, TCNDA will go through the other rest connections to process them one by one. Because connected-neighbors list is empty until now, both nodes (6) and (7) will be added to it due to processing connection (6 < > 7) . No nodes will be added after processing connections (0 < > 1) ,(0 < > 3) and (0 < > 2) because node (6) and (7) are not common with these connections . After that, node (3) and (1)will be add after processing connections (3 < > 7) and (1 < > 7) because node 7 is common in them. In this moment iteration 1 is finished by existence of node 6,7,3,1 in connected-neighbors list of tested nodes (9). At second iteration, node (0) will be added when processing connection (0 < > 1) because now node (1) is common in this connection. Similarly, node (2) will be added to this list during processing of connection (0 < > 2) in the third iteration. Finally at fourth iteration, there are no nodes that will be added, so process of building node (9) connected neighbors list is complete. We should notice that number of iterations will be changed according to the tested node. As shown in table 3. We can see

the size of the node (9) connected neighbors list is equal to the number of k-hop neighbors. So, node (9) is not critical alone.

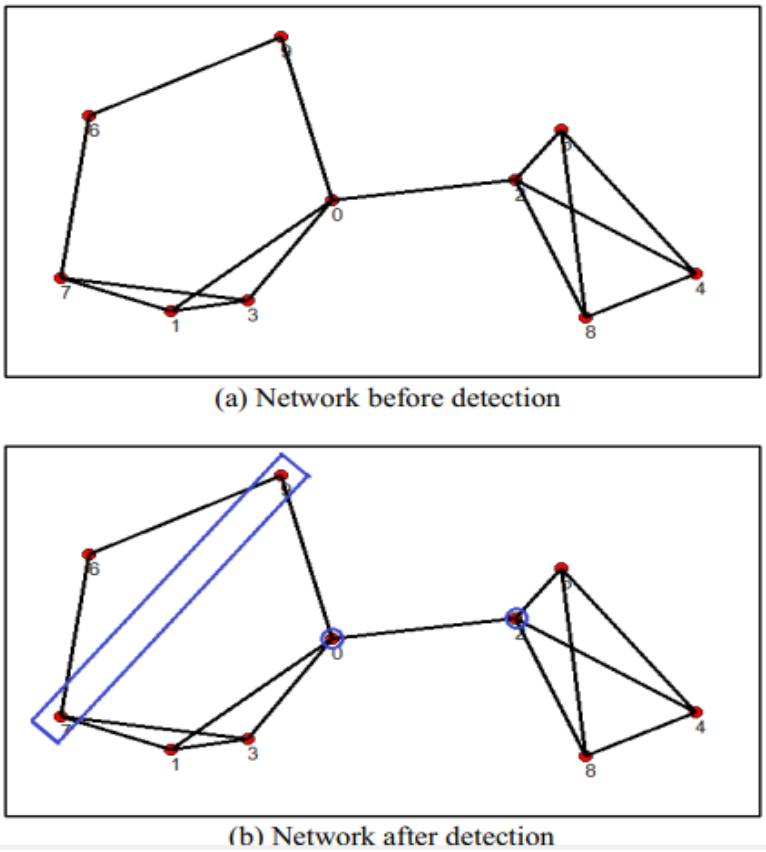

Figure 2: example of network and the detection of criticality using TCNDA.

If the node is not critical then we will test every node in k-hop neighbors to see if there is another node with the node we test and both of them together are critical. If the node is leaf node or critical node ignore it, on the other side we will process the connection list one by one, if the connection has the node itself or the node tested in this then ignore the connection and go to the next connection. And continue like before until connected neighbors list filled. If the size of the connected-neighbors list is equal to the number of (k-hop neighbors of the node -1), the node and the tested node together are not critical. But if the size is less, then the node and the tested node together are critical. If both of nodes together are critical, both nodes send a confirmation message to the other node to ask if it also see the same thing. The confirming message increases the accuracy, because this allows the node to know if there is alternative path out of its local k-information.

For example, after node (9) process its connection list of 2-hop and generate connected-neighbors list, and the size of the connected-neighbors list indicates that node (9) is not critical alone. Then node (9) will test every node in its 2-hop neighbors except the critical node already like node(0) and generate the connected-neighbors list to test criticality of node(9) and the tested node together. If the size of the connected-neighbors list is less than

the number of 2-hop neighbors -1then node (9) and the tested node together are critical. From table 3 we can see node (9) are critical with node (7).

Table 3: node (9) lists for 2-hop.

2-hop neighbours	2-hop connection list of node(9)	connected-neighbours list to test criticality of					
		node (9)	node (9) and node (6)	node (9) and node (7)	node (9) and node (1)	node (9) and node (3)	node (9) and node (2)
6	9 < > 6	6	0	0	6	6	6
0	9 < > 0	7	1	1	7	7	7
7	6 < > 7	3	3	3	3	1	3
3	0 < > 1	1	2	2	0	0	1
1	0 < > 3	0	7		2	2	0
2	0 < > 2	2					
	3 < > 7						
	1 < > 7						

TCNDA identify the k-hop criticality of the node from the first time after build the connection list. But if the node is not critical, we can't determine if there is another node from its direct neighbors both of them together are critical from the first time. Because it's dependent of the criticality of the direct neighbors, If the node is leaf or critical don't take it into account. So, this part is determined after each node shares its criticality. Obviously, if a node identified by TCNDA as k-hop not critical it will be globally not critical and determine noncritical node is very important in recovery algorithms, but if a node identified by TCNDA as k-hop critical or two noncritical nodes together are critical it may be not globally critical because it is possible to exist alternative paths in the network out of the range of the k-hop. Taking into account no critical point is lost this kind of localized algorithms is suitable and helpful. Figure 3 shows the pseudo code of the proposed TCNDA.

4. RESULTS AND ANALYSIS

The accuracy of the proposed localized algorithm is proved by comparing the results with the equivalent globalized algorithm which are perfectly accurate through simulation experiments. A simulator coded in C# is developed to validate the performance of proposed algorithm. The algorithms presented in [13],[14] are used to generate random connected topologies with specified radio range (R) OR average number of neighbors for each node (D). The experiments are accomplished in area (A) with dimensions equal to 800×800 m. The radio range (R) of the nodes and the number of nodes is changed among 10, 30, 50, 70, 90 and 110.Also, the average number of neighbors for each node (D) is changed among (3 to 9). For each experiment, the average value of the results is taken after executing each algorithm 10 times.

Figure 4 shows the accuracy of detection of critical pairs when n=50 and d vary between (3 to 9) in (A), when n=50 and R vary between (10 to 110) in (B) and when d=4 and N vary between (10 to 110) in (C). The performance of TCNDA in detection of critical pairs with local information is very competitive compared with the global information that can exactly detect all critical pairs. The accuracy of TCNDA in our experiment for 2-hop and 3-hop local information is generally over 96% and the main reason for this high accuracy comparing to

```
If Connection List Size = 1 then
    node N is not critical
else if Connection List Size > 1 then
    lContinue = true
    While (lContinue == True)
        lContinue = false
        For each connection in the connections List
            If connection don't contain N then
                If connected-neighbours list size = 0 then
                  Add the pair of current connection into the connected-neighbours list
                  lContinue = true
                Else
                   If any node from the pair of connection exist in connected-neighbours list then
                       If the other node not exist in connected-neighbours list then
                           Add the other node to connected-neighbours list
                           lContinue = true
                       End if
                   End if
                End if
            End if
         End for
     End while
     If connected-neighbours size < number of k-hop neighbors then
        Node N is critical
     Else
        Node N is not critical
        Clear items in connected-neighbours list
        For each node M  in  k-hop neighbors
           If m is not ( leaf node or critical node ) then
              lContinue == True
              While (lContinue == True)
                 lContinue = false
                 For each connection in the connections List
                    If connection don't contain N or M then
                       If connected-neighbours list size = 0 then
                         Add the pair of current connection into the connected-neighbours list
                         lContinue = true
                       Else
                          If any node from the pair of connection exist in connected-neighbours list then
                              If the other node not exist in connected-neighbours list then
                                  Add the other node to connected-neighbours list
                                  lContinue = true
                              End if
                          End if
                       End if
                    End if
                 End for
              End while
           End if
           If connected-neighbours size < number of k-hop neighbors -1 then
              Send confirm message to node  m if received yes Add node M to critical together list
           End if
        End for
    End if
End if
```

Figure 3: Pseudo code of the proposed TCNDA of each node N in the network

the accuracy of the detection of one critical node presented in [11],[12] is the confirming message which node sent to other node when it detect its critical with it, Because this allow the node to know if there is alternative path out of its local k-information. So, the error of determining pair of nodes as local critical which are indeed not is decreased.

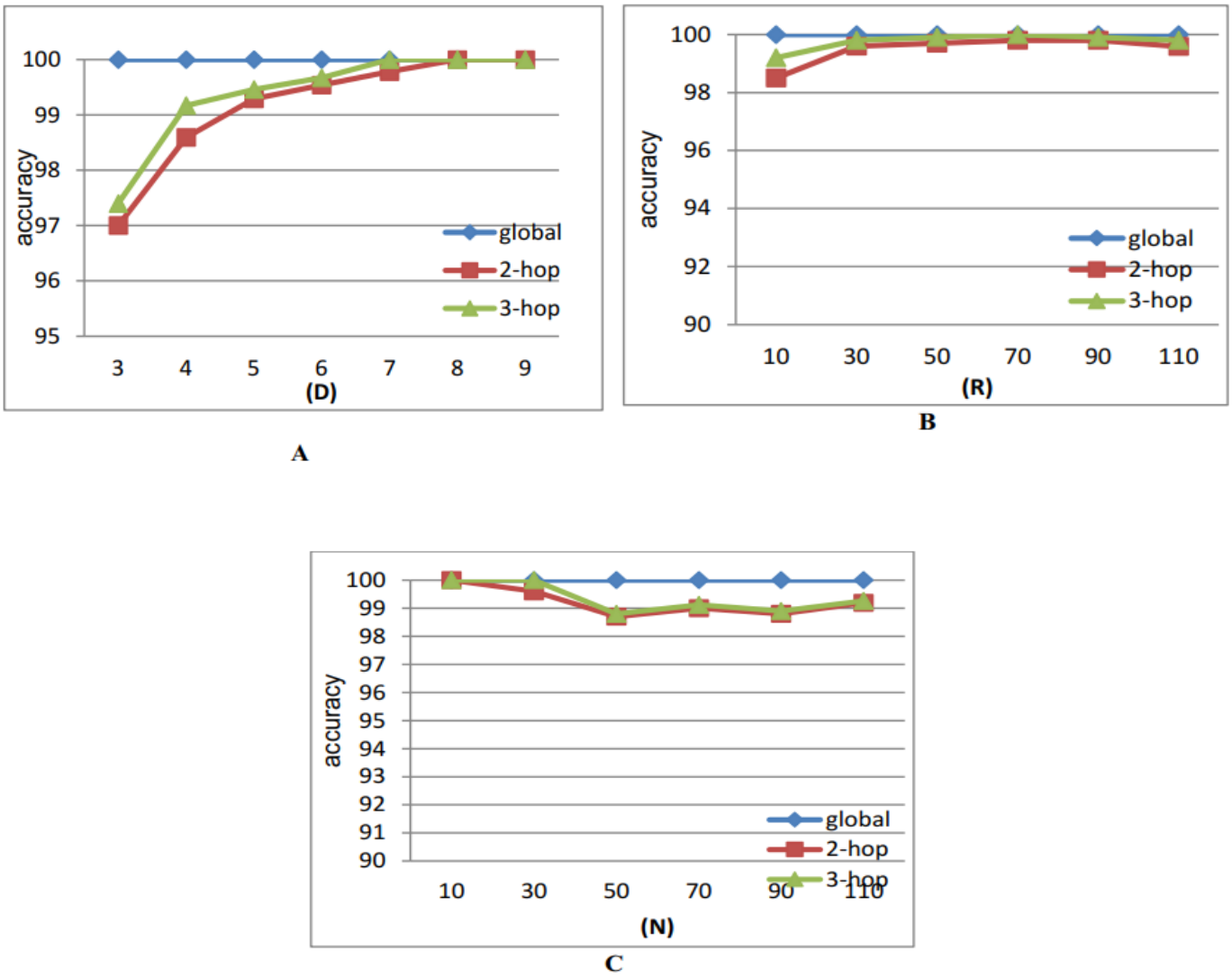

Figure 4: the accuracy of detection of critical pairs when changing in d (A), when changing in radio range (B) and when changing in number of nodes (C).

Table 4 shows the number of critical nodes and number of second critical nodes which are not critical alone but critical with other node two together, the number of nodes used in experiments is 30 and the radio range is 50. The table shows that when the number of critical nodes is increased, the number of 2nd critical nodes is decrease. Some pairs detected by local information not detected by the global information and this is understandable because alternative paths may be exist but the local information of both nodes create the pair

critical not contain this alternative paths. Also, some pairs detected by global information not detected by the local information because the node may be local critical and not global critical, so the node not calculated as second local critical but its already detected as local critical, so this error is acceptable in most scenarios.

Table 4: the number of critical nodes and 2nd critical nodes when n=30 and R=50.

	Global critical	2-hop critical	3-hop critical	Global 2nd critical	2-hop 2nd critical	3-hop 2nd critical
1	12	20	17	8	0	4
2	17	17	17	0	0	0
3	14	17	14	3	2	3
4	12	21	12	9	0	9
5	13	22	19	9	0	3
6	15	19	15	4	0	4
7	13	18	13	9	4	9
8	16	16	16	2	2	2
9	18	18	18	0	0	0
10	16	16	16	3	3	3

5. CONCLUSION

In this paper, TCNDA algorithm is developed for detection of critical node in ad hoc networks. It is localized and distributed algorithm. This algorithm enables each node to determine if it's local critical or there are another node from its k-hop neighbors and both of them together are critical. TCNDA is evaluated and the effectiveness of it in determining the criticality of the node is validated through simulation experiments. Simulation results demonstrate that the performance of TCNDA with local k-hop information is quite competitive compared to centralized algorithm with global network information.

REFERENCES

[1] I. Ahmad, U. Ashraf, and A. Ghafoor, “A comparative QoS survey of mobile ad hoc network routing protocols,” Journal of the Chinese Institute of Engineers, vol. 39, no. 5, pp. 585–592, 2016.

[2] M. Rath, B. K. Pattanayak, and B. Pati, “Energy Efficient MANET Protocol Using Cross Layer Design for Military Applications,” Defence Science Journal Def. Sc. Jl., vol. 66, no. 2, p. 146, 2016.

[3] S. Kaur, S. Kaur, and C. Sharma. “An Overview of Mobile Ad hoc Network: Application, Challenges and Comparison of Routing Protocols,” IOSR-JCE IOSR Journal of Computer Engineering, vol. 11, no. 5, pp. 7–11, 2013.

[4] Jain and S. Chand, “Issues and Challenges in Node Connectivity in Mobile Ad Hoc Networks: A Holistic Review,” Wireless Engineering and Technology WET, vol. 07, no. 01, pp. 24–35, 2015.

[5] M. Edalatmanesh." Heuristics for the Critical Node Detection Problem in Large Complex Networks",PhD thesis, Faculty of Mathematics and Science, Brock University, St. Catharines, Ontario, 2013.

[6] M. Duque-Anton, F. Bruyaux, and P. Semal, “Measuring the Survivability of a Network: Connectivity and Rest Connectivity,” Eur. Trans. Telecomm. European Transactions on Telecommunications, vol. 11, no. 2, pp. 149–159, 2000.

[7] D. Goyal and J. J. Caffery, "Partitioning Avoidance in Mobile Ad Hoc Networks Using Network Survivability Concepts," in the Proceedings of the 7th International Symposium on Computers and Communications (ISCC'02), Giardini Naxos, Italy, 2002, pp. 553 – 558.

[8] S. Xiong and J. Li, "An Efficient Algorithm for Cut Vertex Detection in Wireless Sensor Networks," in the proceedings of the 2010 IEEE 30th International Conference on Distributed Computing Systems (ICDCS) Genova, Italy, 2010, pp. 368 377.

[9] X. Liu, L. Xiao, A. Kreling, and Y. Liu, "Optimizing overlay topology by reducing cut vertices," in in the Proceedings of the 2006 international workshop on Network and operating systems support for digital audio and video, Rhode Island, USA, 2006.

[10] M. Sheng, J. Li, and Y. Shi, “Critical Node Detection in Mobile Ad Hoc Network”, the 20th International Conference on Advanced Information Networking and Applications (AINA’06), 2006.

[11] M. Hauspie, “Localized Algorithms for Detection of Critical Nodes and Links for Connectivity in Ad hoc Networks”, in: "Proc. 3rd IFIP Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET 2004), Bodrum, Turkey", 2004.

[12] M. Imran, M. A. Alnuem, M. S. Fayed, and A. Alamri, “Localized Algorithm for Segregation of Critical/Non-critical Nodes in Mobile Ad Hoc and Sensor Networks,” Procedia Computer Science, vol. 19, pp. 1167–1172, 2013.

[13] Furuzan Atay, and Ivan Stojmenovic. "Generating Random Graphs for Wireless Actuator Networks." IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, 2007.

[14] Ibrahim S. Fayed, Mokhtar A. A. Mohamed, Gamal Attiya and Nawal A. El-Fishawy, " Two Efficient Approaches for Generating Topologies of Ad-hoc Networks", in proc. of 1st internationsl conference on Advanced Technology and Applied Sciences (ICaTAS), Kuala Lumpur, MALAYSIA, 2016.