Localized Algorithms for Detection of Critical Nodes and Links for Connectivity in Ad hoc Networks

Milenko Jorgic, Michaël Hauspie, David Simplot-Ryl, Ivan Stojmenovic

Localized algorithms for detection of critical nodes and links for connectivity in ad hoc networks

Milenko Jorgić, Ivan Stojmenović
SITE, University of Ottawa, Ottawa,
Ontario K1N 6N5, Canada
{mjorgic, ivan}@site.uottawa.ca

Michaël Hauspie, David Simplot-Ryl
IRCICA/LIFL, Univ. Lille 1, INRIA futurs, France
{Michael.Hauspie, David.Simplot}@lifl.fr

Abstract

Ad hoc network normally has critical connectivity properties before partitioning. The timely recognition is important in order to perform some data or service replication. Several existing centralized or globalized algorithms declare an edge or a node as critical if their removal will separate the network into several components. We introduce several localized definitions of critical nodes and critical links, using topological or positional information. A node is critical if the subgraph of k-hop neighbours of node (without the node itself) is disconnected. We propose three definitions of critical links, based on verifying common k-hop neighbours, loop length, and critical status of link endpoints, respectively. The experiments with random unit graph model of ad hoc networks show high correspondence of local and global decisions. For instance, in experiments with 500 nodes in connected random unit graphs, over half of locally estimated critical nodes and links were indeed globally critical even for k=1 (the accuracy increases to over 70% for k=2 and over 80% for k=3), for average number of neighbours ranging from 3 to 15. The errors mostly occur when alternative routes exist but are relatively long, and therefore may not provide satisfactory service in applications. Therefore our localized protocols provide faster and often more reliable partition warnings for possible timely replication decisions.

1. Introduction

An ad hoc network is a collection of wireless mobile hosts forming a temporary network without the aid of any fixed infrastructure. They have potential application in civilian and military environments such as disaster relief, conference, wireless office, and battlefield. Ad hoc sensor networks for monitoring environment are also being deployed.

In an ad hoc network a message sent by a node reaches all its neighbouring nodes that are located at distances up to the transmission radius. A widely accepted basic graph-theoretical model for ad hoc networks is a *unit graph* model, defined in the following way. Two nodes *A* and *B* in the network are neighbours (thus joined by an edge) if and only if the Euclidean distance between their coordinates in the network is at most *R*, where *R* is the transmission radius which is equal for all nodes in the network. Due to the limited transmission radius, the routes between two nodes are usually created through several hops.

A node or link is critical if its removal will disconnect the graph into two (or more) separate components. Suppose we have a user *E* and a server *L* (see Figure 1.). When user *E* requests a service or data from *L*, the request follows the path *ECBAIJL*. In this example, nodes *A, B* and *J,* and link *AB,* are critical, since removal (or movement) of any of them will partition the network. Node *E* may initiate some actions, such as replicating data from *L* in its own memory, or look for alternative service in different part of network, before it is too late.

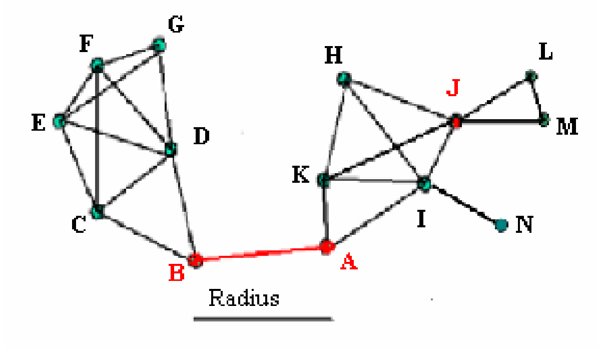

Figure 1. Unit graph representation of multi-hop wireless network

DFS (depth first search algorithm) was used to detect critical links in [DBS, T, GC]. It is a centralized algorithm, and can be also implemented in globalized distributed manner. A centralized algorithm requires that a node should be aware of global topology. In practice, this method is inefficient and involves a quadratic (in number of nodes) communication overhead in order to update link information when nodes are moving, or when nodes change their status from active to sleeping and vice versa. In a globalized distributed implementation, *DFS* can be performed in the network without global knowledge at any node, but with memorization at nodes. In [GC] once critical links in an ad hoc network are detected, two ways are proposed to delay or avoid their failure: changing the trajectory of one or both nodes forming the critical link and bringing another node to reinforce the link. Increased delivery rates were reported due to prolonged network connectivity. However, the communication overhead due to running *DFS* for detecting critical links was not measured. In this article, we propose to apply localized algorithms instead of globalized ones for partition detection. Localized algorithms are distributed in nature and resemble greedy algorithms, where simple local behaviour achieves a desired global objective. In a localized algorithm for detection of critical links each node makes a decision to determine critical nodes or links based only on limited local knowledge.

The purpose of this article is to show that, with high probability, possible partition can be detected by localized algorithms, therefore greatly reducing communication overhead and the speed of detecting, allowing network to replicate data or service in time if needed. The paper is organized as follows. We describe the existing work on critical links and node detection in Section 2. In Section 3 we propose and evaluate localized algorithms for detection of critical links/nodes and compare our localized algorithms with the known glob orithm. Finally, we conclude our work in Section 4.

2. Literature Review

Critical nodes and links are only considered for a connected graph (or separately for each connected component of a graph). A node *A* is critical if its removal will disconnect the graph into two components. A straightforward algorithm for detecting critical nodes may consider, for each node *A*, the subgraph obtained by its removal and removal of all its adjacent edges, and testing whether this subgraph is connected. If it is not connected, the corresponding node *A* is a critical node. As a consequence, node that has only one neighbour is not critical (e.g. node *N* in Fig. 1).

A faster global algorithm for detecting critical nodes was described by Duque-Anton, Bruyaux, and Semal [DBS], Tarjan [T], and Goyal and Caffery [GC] who used *DFS*. It is a centralized algorithm which can be also implemented in globalized distributed manner. While executing *DFS* on an undirected graph, we start at an arbitrarily chosen node which becomes the root. We keep traversing fresh edges and mark nodes as “visited”; on the way we keep pushing nodes into a stack data structure. This process is continued until we reach a node which is only connected to already visited nodes. At this point we keep backtracking up to a vertex which has edges connecting them to nodes which have hitherto not been visited. With a little thought it can be seen that such a node will always be a critical node of the graph. Alongside the identification of the critical node, it is easy to pop the downstream nodes from the stack into a set which corresponds to a bi-connected component. Since the above steps can be executed during *DFS* in the same pass, identification of critical nodes takes only linear time.

Critical links can de defined in several ways. One possible definition is that a link *AB* is critical if both endpoints *A* and *B* are critical nodes. However, two critical nodes may have alternate path between them. For example, in Fig. 4, nodes *O* and *Q* are critical, but alternate path between them exits via node *P*. It is therefore better to define critical link as the link connecting two critical nodes so that, when this link is eliminated from the graph, the graph becomes disconnected.

In [GC], once critical links in an ad hoc network are detected, two ways are proposed to delay or avoid their failure: changing the trajectory of one or both nodes forming the critical link and bringing another node to reinforce the link. Increased delivery rates were reported due to prolonged network connectivity. However, the communication overhead due to running *DFS* for detecting critical links was not measured. Karumanchi, Muralidharan, and Prakash [KMP], Li, and Rus [LR], Vahadat and Becker [VB], Park and Corson [PC] attempt to improve the communication without avoiding or delaying partitioning ('post-partitioning' approaches).

Hara [H] proposed three replica allocation methods to improve data accessibility by replicating data items on mobile hosts. The first method is to make a lot of replicas at each node, while the two others begin with a step in which each mobile host periodically broadcasts its host identifier and information on access frequencies to data items. After all mobile hosts complete their broadcasts, every host knows its connected mobile hosts. This partition detection method clearly requires a lot of communication overhead.

Wang and Li [WL1, WL2] proposed a mechanism to allow servers in an ad hoc network to detect the future partitions and to replicate them in each predicted partition. However, this solution uses a strong centralized approach to detect partitions (their algorithm captures the network mobility information using pattern recognition technique in the velocity space, so that the time and location of network partitioning can be predicted) and its applicability as such may be questionable.

Hauspie, Simplot and Carle [HSC] proposed to evaluate stability of a path from a source to destination by a function that depends on disjoint path between them, and the hop distance of each of these paths. When the function reaches a threshold, data or service replication is performed. The protocol has a significant communication overhead for evaluating the function.

Koskinen [K] examined critical transmission ranges for biconnectivity and triconnectivity of ad hoc networks. He experimentally established an asymptotic behaviour that these types of critical links can be determined by a function which is square root of the ratio of area and linear function of number of nodes (coefficients of that function are parameters to be determined).

Hajiaghayi, Immorlica, and Mirrokni [HIM] considered the problem of assigning transmission power to nodes so that the sum of powers is minimized and the network is *k*-connected. They use energy cost d^{α} for transmission between two nodes at distance d (α is a constant between 2 and 6) and the algorithms are globalized. To guarantee *k*-connectivity, [BHM] uses a solution where most nodes need to have *3k* or more neighbors, by enforcing minimal angle between two selected neighbors.

Shah, Chen, and Nahrstedt [SCN] aimed at enhancing data access in an ad hoc network by detecting partitions in it. They propose a data replication mechanism based on partition detection for allowing one node to access the data from another node even if the connection is physically broken. Each node of a connected group knows the behavior of other members in a group, because each node embeds a positioning system (GPS) by successive measures computes its velocity, and spreads that information to the other nodes. Due to this information, it is possible to predict when a node will leave its group. The 'node-to-leave' picks another node of the group to be a host of the data and performs a data replication on that node. The main advantage of this method is that each node knows exactly when the partition occurs (regular node movement is assumed, without sudden changes in direction). However, this method has two main disadvantages. First, a positioning system is needed. Second, the network is continuously and relatively highly loaded due to information exchange among nodes. While general ideas in [SCN] are good, the essential details are missing. The definition of group of nodes is not given, and one can even assume the whole connected network to be a group, since it also satisfies the vague definition given. The protocol for predicting link breakage, based on predicting future locations and connectivity of two nodes, is described in detail (a similar protocol was described earlier by Stojmenovic, Russell and Vukojevic [SRV]) but there is no description of any protocol to detect group partitioning; it was left up to a node to decide without giving details on how it is actually decided.

Li and Rus [LR] propose an approach in which nodes actively modify their trajectories to transmit messages. They develop algorithm that minimize the trajectory modifications under two assumptions: the movements of all the nodes in the system are known and not known, respectively.

In cooperative caching, discussed by Cao, Yin and Das [CYD], data from server are replicated on some nodes in ad hoc network so that access demands by other nodes can be satisfied by replicated files rather than original files, which should reduce traffic in the network or even provide service if the server becomes disconnected in the meanwhile. The described methods include caching data paths toward replicated copy by current node, or making another copy of data at the node, plus some hybrid method based on some criteria.

3 Localized algorithms for partition detection

Distributed and dynamic nature of the ad-hoc networks requires the design of localized algorithms to address scalability, robustness and energy efficiency issues. Localized algorithms that we have developed discover all critical nodes and links very quickly. However, these algorithms may detect some nodes and links as critical although they may not be globally critical. This is unavoidable since local knowledge only is used, therefore with such restriction it is impossible for a node to learn about alternate connections in different parts of the network. On the other hand, in applications, often long alternate paths do not provide satisfactory service, thus localized method may even provide more useful decision. Moreover, the partition detection is done faster and with much less communication overhead.

We first define what is the local knowledge available to nodes and how nodes gain it. We use the notion of *k*-hop knowledge. Two nodes are considered to be *k*-hop neighbours if and only if the shortest route between them has *k* or less hops. Awareness of itself only is represented as 0-hop knowledge. This may or may not include geographic position of the node. Localized algorithms that use position information can only be applied on nodes that are equipped with GPS or find their relative coordinates by measuring signal strengths or time delays in mutual communication. Nodes collect *k*-hop knowledge by sending 'hello' messages to its neighbours containing the graph of their (*k-1*)-hop neighbours. Thus 1-hop knowledge is a list of direct neighbours, with or without their geographic positions. We refer to these cases as being *topological* and *positional* information, and corresponding knowledge as being *k-hop topological* or *k-hop positional* information, respectively. The 2-hop topological information is obtained by transmitting lists of 1-hop neighbours, and the subgraph of 2-hop neighbours therefore includes existing links between 1-hop neighbours, and between a 1-hop and a 2-hop neighbours, but not possible links between 2-hop neighbours. On the other hand, 2-hop positional information includes such links and information, since node learns the position of 2-hop neighbours and may, based on distances between them and the unit graph used, decide whether or not they are neighbours. Generalizing this, *k*-hop topological information, and corresponding subgraph of *k*-hop neighbours, include all existing links between a *k*-hop and a (*k-1*)-hop neighbours, but not information on whether or not two strictly *k*-hop neighbours (that is, two nodes which are *k*-hop but are not (*k-1*)-hop neighbours) are connected. Since the information about any node in the positional case includes its position, in case of *k*-hop positional information, such information is additionally available. Therefore, localized algorithms with position information have more information than localized algorithms with topological information, and are consequently more accurate; they discover less falsely detected critical links and nodes. Furthermore, if a node/link is declared critical by a local algorithm that uses position information, it is also declared as critical by the localized algorithm that uses the corresponding topological information. The reason is that the graph may only have fewer edges and thus the partition detected by positional information cannot be 'sealed' by the corresponding topological information.

We shall now describe our localized algorithms for detecting critical nodes and links. In each of topological and positional cases, we give one definition of critical nodes and three definitions of critical links. The three definitions are based on verifying common neighbours, loop length, and critical status of link endpoints, respectively.

3.1 Localized algorithms for detection of critical nodes

For each node *A*, consider subgraph of *k*-hop neighbours of *A*, where *A* and all its incident edges are excluded. In case of positional information, two nodes in that graph are connected if they are connected in the original graph. In case of topological information, two nodes in that graph are connected if they are connected in the original unit graph, and at least one of them is (*k-1*)-hop neighbour of *A*.

A node *A* is *k*-critical node if the corresponding subgraph of *k*-hop neighbours of *A* is disconnected. Based on information used, this is further specified as being topologically or positionally *k*-hop critical node. The corresponding algorithms are referred to as being *k-top_critical_node* and *k-pos_critical_node algorithms.*

Clearly, if a node is globally critical, localized algorithm will detect it as such. Further, if a node is not declared as critical by *k-top_critical_node* algorithm it is also not declared as critical by *k-pos_critical_node* algorithm. That is, if a node is declared as critical by *k-pos_critical_node* algorithm it is also declared critical by *k-top_critical_node* algorithm.

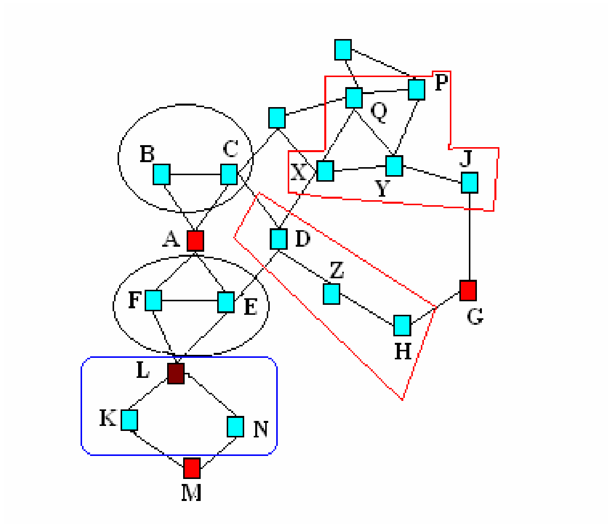

Figure 2. Node *A* is 1-hop critical, node *G* is topologically 3-hop critical and positionally 2-critical, node *M* is 1-hop critical

We will now discuss particular cases and give some examples as illustration. For *k=1,* if topological information is used, no links between neighbours exist in the decision graph, therefore all nodes are declared critical with *1-top_critical_node* algorithm. This is obviously not very helpful.

Figure 2 illustrates the localized definitions of critical nodes. 1-hop neighbours of *A* can be divided into two sets *{B, C}* and *{E,F}* (circled in Fig. 2) which are disconnected. Therefore *A* is positionally (and therefore topologically) 1-hop critical. Node *A* is not 2-hop critical since its 2-hop neighbour *D* will connect two sets. Node *M* is 1-hop critical since its 1-hop neighbours *K* and *N* are not connected. However, it is not 2-hop critical since its two hop neighbours *K, N,* and *L* create a connected subgraph (marked by a rectangle in Fig. 2), both topologically and positionally. Node *G* is topologically 3-hop critical since its 3-hop neighbours are divided in two subgraphs with vertices *{H, Z, D}* and *{J, Y, X, Q, P}* (the two sets are enclosed by polygons in Fig. 2) which are disjoint. However, when position information is added, node *G* can recognize that *X* and *D* are in fact neighbors, and that two subgraphs are in fact connected, therefore *X* is not positionally 3-hop critical.

3.2 Localized algorithm for detection of critical links based on common *k*-hop neighbours

If topological information is used, the algorithm, referred to as *k-link_top_critical* algorithm, applies the following criterion. *AB* is a critical link if the sets of *k*-hop neighbours of *A* and *B* (assuming that the link *AB* does not exist) are disjoint. For *k=1,* this reduced to the following *1-link_top_critical* algorithm: *AB* is critical link if *A* and *B* have no common neighbours (that is, there is no node *C* so that both *AC* and *BC* are in the unit graph).

If position information is used, the corresponding *k-link_pos_critical* algorithm is defined as follows. *AB* is a critical link if the sets of *k*-hop neighbours of *A* and *B* (assuming again that the link *AB* is removed first from the graph) are disjoint, and there are no two nodes, one from each set, which are neighbours. For *k=1,* the criterion is that *A* and *B* have no common neighbours that are within transmission range of each other (that is, neighbours).

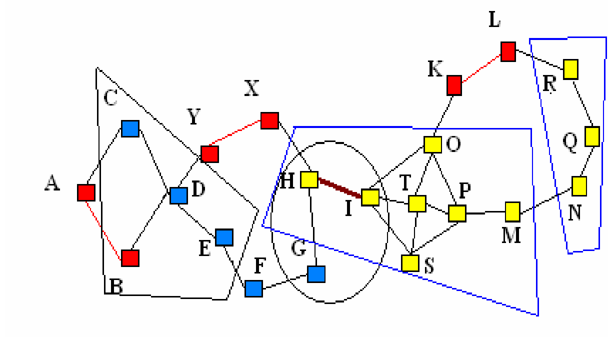

Figure 3. Examples of critical links *HI, AB, XY* and *KL*

Consider the example graph in Figure 3. It has only one globally critical link, *HI.* Localized algorithms will detect some more critical links. For example, *1-top_critical_link* algorithm declares the link *AB* as critical, because nodes *A* and *B* have disjoint 1-hop neighbours (*C* and *D*). However, *1-pos_critical_link* algorithm declares correctly that the link *AB* is not critical, because *A* and *B* may together agree that *C* and *D* are in fact neighbours, therefore there exists an alternative path from *A* to *B* which does contain the link *AB.* The link *XY* is declared as critical with both 2-*top_critical_link* and 2-*pos_critical_link* algorithms. The 2 hop neighbours of *X* (*H, G, I,* circled in Fig. 3) and *Y* (*D, C, B, E,* enclosed in a quadrilateral in Fig. 3), are disjoint and there are no two nodes, one from each set, which are neighbours, hence the link is declared as critical. Nevertheless, the *3-top_critical_link* scheme correctly declares the link *XY* as not being critical because the 3-hop neighbors of *X* (*H, I, O, T, S,G, F*) and *Y* (*D, C, A, B, E, F*) are not disjoint. If we examine the link *KL* with *3-top_critical_link* algorithm, we declare it as critical, because the 3-hop neighbors of *K* (*O, P, M, T, S, I, H*) and *L* (*R, Q, N*), both enclosed in quadrilaterals in Fig.3, are disjoint. On the other hand, 3-*pos_critical_link* algorithm correctly detects that the link *KL* is not critical. Even though the neighbors of *K* and *L* are disjoint, the algorithm detects that the link between *M* and *N* (which are 3-hop neighbors of *K* and *L* respectively) exists and declares the link *KL* as not being critical.

3.3. Localized algorithms for detection of critical links based on loop length

In this section, we introduce critical links definitions which are based on finding the length of shortest loop between two link endpoints. A link *UV* is *k-loop_critical* if the hop distance between *U* and *V* in the given graph, with only edge *UV* being eliminated, is *>k.* There are several possible implementations of this definition, since a common decision should be made between two nodes, link endpoints. In general, we can consider k_U-hop neighborhood of *U* and k_V-hop neighbourhood of *V,* $k_U + k_V = k$. If topological information is used, the algorithm is defined as follows. A link *UV* is *(k_U,k_V)-loop_top_critical* if the sets of k_U -hop neighbors of *U* and k_V -hop neighbors of V are disjoint. It follows then that a link is *k-link_top_critical* if and only if it is *(k,k)-loop_top_critical* (or *2k-loop_critical* in a more general definition). Because of such equivalency, and expectation that $k_U = k_V$ is reasonable to assume, we did not implement this definition. If positional information is used, the corresponding algorithm applies the following test. A link *UV* is *(k_U,k_V)-loop_pos_critical* if the sets of k_U -hop neighbors of *U* and k_V -hop neighbors of *V* are disjoint, and there are no two nodes, one from each of these sets, that are neighbours. It follows then that a link is *k-link_pos_critical* if and only if it is *(k,k)-loop_pos_critical* (that is, *2k-loop_critical*). Note that our initial definition of a *k-loop_critical* link is equivalent to the new definition of *(k,0)-loop_critical* link.

Figure 4 illustrates these definitions. For example, after applying *1- and 2-loop_top_critical* algorithms on link *AB*, we declare the link as critical because from node *A* we can not reach the node *B* with 2 hops. However, the link will not be detected as critical with *2-loop_pos_critical* algorithm, since 2-hop neighbour node *D* of *A* is neighbour of node *B*, based on their geographical positions. The *3-loop_top_critical* algorithm detects the link *XY* as critical, but the *3-loop_position_critical* correctly detects the link as not being critical, since 3-hop neighbour *F* of node *Y* is a neighbour of *X*. The link *JK* is the link where *3-loop_position_critical* algorithm wrongly declares it as critical. The loop is too wide, and more hops are needed in order to correctly declare the link as not being critical. The *1-loop_top_critical* scheme is very week, which is illustrated on link *MN*. This scheme only detects the node *O*, which obviously does not form a loop from *M* to *N*, therefore, all the link is declared critical with *1-loop_top_critical* algorithm. However, the *1-loop_pos_algorithm* correctly declares the link *MN* as not being critical.

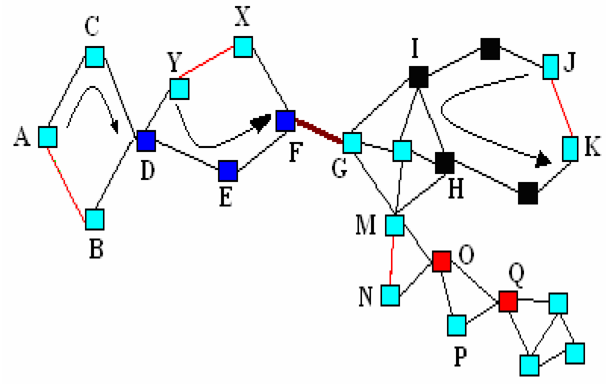

Figure 4. Loop length based critical links *AB, XY, MN* and *JK*

Localized algorithms for detection of critical links based on loop length, falsely declare many links as critical.

3.4 Localized algorithms for detection of critical links based on critical nodes

Perhaps the simplest definition of critical nodes is by using prior recognition of critical nodes, as follows. A link *AB* is declared *k-pos_link-by-node_critical* (*k-top_link-by-node_critical*) if both *A* and *B* are declared as *k-pos_critical nodes* (*k-top_critical nodes,* respectively). The advantage of this definition over other two is that the implementation of it does not require two nodes to exchange their neighbourhood information beyond already exchanged one. Instead, they need to exchange merely their decisions about being critical nodes.

Alternatively, each node may decide which of its links are *k*-hop critical by using its (*k+1*)-hop information, which involved more communication overhead for collecting than the suggested decision exchange method. The communication overhead involved with detection of critical links via critical nodes is the smallest among the mentioned definitions and corresponding implementations. However, as already observed, this algorithm is not as accurate for detection of critical links as *k-top/pos_critical_link algorithm*. Two critical nodes may be connected with more than one link. From Figure 4 we see that nodes *O* and *Q* are critical nodes and that they have a direct link between them, yet they also have a common neighbour *P*, which creates a path *OPQ*. Therefore, the link *OQ* is in fact not critical, which can be easily verified with *1-top_critical_link* algorithm. These cases are present in our simulations; hence, due to the inaccuracy of this algorithm, we did not perform the experiments for it.

4. Performance evaluation

We have measured the accuracy of proposed localized critical nodes and links detection algorithms by comparing them with the corresponding globalized algorithm, that is, with the correct conclusion. We will describe here the experimental results obtained by experimenting with connected random unit graphs. The master thesis of the first author (in preparation) contains also data for random unit graphs which are allowed to be disconnected.

We generate connected random unit graphs with *n* nodes and desired density (average number of neighbors) *d* using the following method. This method is selected since we wanted to estimate the performance also for very sparse networks. This was a time consuming process with alternative methods (e.g. generating each node at random, deciding proper transmission radius, testing connectivity at end).

An approximate radius *r* is obtained from the formula $d=(n-1)*r*r*\pi/(a*a)$. The first node is randomly generated (that is, its *x*- and *y*-coordinates are chosen at random) in a given square with edge length *a*. Each of following *n-1* nodes is generated repeatedly, at random, and tested whether it is within distance *r* to at least one of previously generated and accepted nodes, until the test is satisfied. Otherwise (when it is at distance $>r$ to all previously accepted nodes), the node is rejected and another node generated and tested instead.

After selecting *n* nodes with this procedure, a connected random unit graph is generated. However, its average degree *q* is not necessarily the desired one, *d*. We now find the exact average degree *d* of generated graph, by counting edges and compare it with desired value *d*. If $q<d$, more edges need to be added, and graph remains connected. We sort all $n*(n-1)/2$ possible edges and desired radius *R* is the $n*d/2$-th edge in sorted list. Unit graph is then decided using this value of *R* (that is, two nodes are neighbours if and only if the distance between them is at most *R*). If $q=d$ then the graph remains unchanged. If $q>d$, this obtained random unit graph is too dense, and some edges need to be deleted by reducing transmission radius. We sort all $n*(n-1)/2$ possible edges in increasing order and find the $n*d/2$-th edge in the sorted list. We use this edge as the transmission radius *R*, and define the corresponding graph, which may not be connected. Dijkstra's *shortest path algorithm* is used to check the connectivity of this graph. If the graph is not connected, it is ignored, and the procedure is repeated. If it is connected, which should happen with high probability with this procedure, the graph is accepted.

Our experiments are performed with *n=100* and *n=500* nodes for several densities, ranging from 3 to 15. In each case, the main measure considered is the *detection ratio*, which is the probability that a node or link declared as critical by considered localized algorithm is indeed critical when verified by global algorithm. We also measured the average number of critical nodes and links detected in the network. We detection ratios for *n=500,* which are overall somewhat better (by as much as 10%) than for *n=100,* which was counterintuitive but encouraging for the scalability of our approach.

4.1 Localized algorithms for detection of critical nodes

Tables 1 and 2 show the detection ratios for critical nodes, using *k-node_top_critical* and *k-node_pos_critical* algorithms, for *k*=1, 2 and 3, obtained after 20 simulations for different values of *d* for random connected unit graphs with n=500 nodes.

Detection ratios are generally over 50%, meaning that over half of locally estimated critical nodes were indeed globally critical even for *k=1* (the accuracy increases to over 70% for *k=2* and over 80% for *k=3*), for average number of neighbours ranging from 3 to 15. As expected the *k-node_pos_critical* algorithm performs better than the *k-node_top_critical* algorithm.

Average degree of neighbours (d)	**Detection Ratio** 2-hop algorithms (%)	**Detection Ratio** 3-hop algorithms (%)
15	50.0	71.4
11	64.2	79.5
10	73.0	90.2
9	70.4	97.7
8	73.3	88.7
7	64.8	82.5
6	67.9	83.6
5	71.5	78.6
4	73.5	91.5
3	66.7	86.5

Table 1. Detection ratios for *k-node_top_critical* algorithm on connected graphs with 500 nodes

Average degree of neighbours (d)	**Detection Ratio** 1-hop algorithm (%)	**Detection Ratio** 2-hop algorithms (%)	**Detection Ratio** 3-hop algorithms (%)
15	50.0	71.4	83.3
11	64.2	82.9	94.4
10	70.8	85.2	97.9
9	55.3	74.6	96.8
8	52.1	75.9	90.9
7	64.8	82.5	90.4
6	61.5	78.9	86.2
5	49.3	76.1	85.1
4	51.3	80.4	92.5
3	56.1	69.6	86.5

Table 2. Detection ratios for *k-node_pos_critical* algorithm on connected graphs with 500 nodes.

Tables 3 and 4 present the average numbers of detected critical nodes for each algorithm. The third column shows the average number of critical nodes detected by global algorithms. The fourth and fifth columns show the number of critical nodes detected by local algorithms for *k=2* and *3*.

Average number of neighbours (d)	**Average number of critical nodes** (global alg)	**Average Number of critical nodes** (2-hop alg)	**Average Number of critical nodes** (3-hop alg)
15	3.0	6.2	4.2
11	6.8	10.6	7.8
10	9.2	12.6	10.2
9	8.8	12.5	9.1
8	11.0	15.0	12.4
7	9.4	14.5	11.4
6	11.2	16.5	13.4
5	14.3	20.0	18.2
4	17.2	23.4	18.8
3	19.2	28.8	22.2

Table 3. Average numbers of detected critical nodes by *k-node_top_critical* and global algorithms

Average number of neighbours (d)	**Average number of critical nodes** (global alg)	**Average Number of critical nodes** (1-hop alg)	**Average Number of critical nodes** (2-hop alg)	**Average Number of critical nodes** (3-hop alg)
15	3.0	6.0	4.2	3.6
11	6.8	10.6	8.2	7.2
10	9.2	13.0	10.8	9.4
9	8.8	15.9	11.8	9.0
8	11.0	21.1	14.5	12.1
7	9.4	14.5	11.4	10.4
6	11.2	18.2	14.2	13.0
5	14.3	29.0	18.8	16.8
4	17.2	33.5	21.4	18.6
3	19.2	34.2	27.6	22.2

Table 4. Average numbers of detected critical nodes by *k-node_pos_critical* and global algorithms

It can be observed that there are not many critical nodes in graphs, especially for graphs with medium density and dense graphs. Localized algorithms do not declare too many nodes as critical that are in fact not globally critical.

4.2 Localized algorithms for detection of critical links

Tables 5 and 6 show the detection ratios obtained after 20 simulations on *k-link_top_critical and k-link_pos_critical* algorithms, for connected random unit graphs with *n=500* nodes.

Number of Nodes (n)	**Average degree of neighbours** (d)	**Detection Ratio** 1-hop algorithm(%)	**Detection Ratio** 2-hop algorithms (%)	**Detection Ratio** 3-hop algorithms (%)
100	15	31.3	50.0	71.4
100	11	39.1	64.2	79.5
100	10	50.5	73.0	90.2
100	9	43.8	71.2	100.
100	8	41.7	78.1	90.9
100	7	44.0	67.7	84.6
100	6	49.0	72.9	85.0
100	5	45.9	72.0	80.7
100	4	47.6	76.9	93.0
100	3	48.9	67.4	86.7

Table 5. Detection ratios for *k-link_top_critical* algorithm on connected graphs with 500 nodes

Average degree of neighbours (d)	**Detection Ratio** 1-hop algorithm (%)	**Detection Ratio** 2-hop algorithms (%)	**Detection Ratio** 3-hop algorithms (%)
15	50.0	71.4	83.3
11	64.2	82.9	94.4
10	70.8	85.2	97.9
9	57.5	76.4	100.
8	55.6	76.9	92.6
7	67.7	84.6	93.6
6	64.6	81.0	86.4
5	52.8	77.9	89.3
4	56.3	83.3	97.6
3	57.2	71.7	86.7

Table 6. Detection ratios for *k-link_pos_critical* algorithm on connected graphs with 500 nodes

From these tables we conclude that the localized algorithms give excellent results. In particular, 3-hops localized algorithms have the accuracy greater than 80% for any *d*, while 2-hop localized algorithms have accuracy greater than 70% in most cases. Even 1-hop localized algorithms declare correctly more than 50% of links.

The average number of critical links detected by local *k-link_top_critical/ k-link_pos_critical* algorithms and the number of critical links detected by global algorithm are recorded in Tables 7 and 8, respectively.

Average number of neighbours (d)	**Average Number of links**	**Average number of critical links** (global alg)	**Average Number of critical links** (1-hop alg)	**Average Number of critical links** (2-hop alg)	**Average Number of critical links** (3-hop alg)
15	5420	1.5	4.8	3.0	2.1
11	4523	3.4	8.7	5.3	3.9
10	4306	4.6	9.1	6.3	5.1
9	4039	4.2	9.6	5.9	4.2
8	3785	5.0	12.0	6.4	5.5
7	3365	4.4	10.0	6.5	5.2
6	3015	5.1	10.4	7.0	6.1
5	2785	6.7	14.6	9.3	8.3
4	2555	8.0	16.8	10.4	8.6
3	2115	9.1	18.6	13.5	10.5

Table 7. Average number of detected critical links by *k-link_top_critical* and global algorithms

Average number of neighbours (d)	**Average Number of links**	**Average number of critical links** (global alg)	**Average Number of critical links** (1-hop alg)	**Average Number of critical links** (2-hop alg)	**Average Number of critical links** (3-hop alg)
15	5420	1.5	3.0	2.1	1.8
11	4523	3.4	5.3	4.1	3.6
10	4306	4.6	6.5	5.4	4.7
9	4039	4.2	7.3	5.5	4.2
8	3785	5.0	9.0	6.5	5.4
7	3365	4.4	6.5	5.2	4.7
6	3015	5.1	7.9	6.3	5.9
5	2785	6.7	12.7	8.6	7.5
4	2555	8.0	14.2	9.6	8.2
3	2115	9.1	15.9	12.7	10.5

Table 8._Average number of detected critical links by *k-link_pos_critical* and global algorithms

From the tables we conclude that local algorithms for detection of critical nodes have approximately the same accuracy as the local algorithms for detection of critical links. This behavior is well expected because if a link is declared as critical, then the two nodes making this link are also declared critical. So, we can expect to have at least two times more critical nodes than links. Some critical nodes are not a part of a critical link. The number of these nodes is not high. For example, no critical nodes that are not part of a critical link was found for *d* between 10 and 15.

We will now give some insight in order to explain obtained result. Figure 5 shows one of obtained random connected unit graphs with 500 nodes, using our described procedure. The average density is high, *d*=15. The main problem that lowers the detection ratio of localized algorithms can be associated with creation of ring structures shown in Figure 5. Ring structures (those producing some critical nodes or links are marked by 'R' in Fig. 5) are very common in these graphs and they are the focal problem in detection of critical links/nodes with local algorithms. The critical links detected by *1-top_critical_link* algorithm are drown in black in Fig. 5. Critical nodes detected by *1-top_critical_node* algorithm are drawn in green in

Fig. 5. Rings are often too wide to be detected by 3 hops and most of the nodes/links that are part of a specific ring are then declared as critical. These rings can be detected if we increase the hop count, but then the nature of local algorithms is lost.

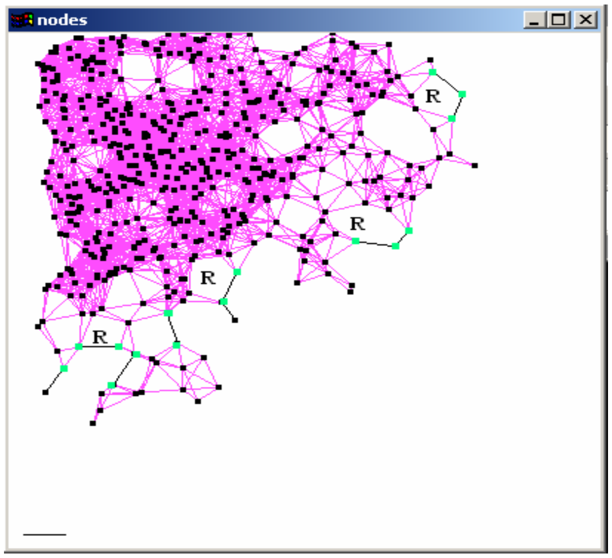

Figure 5. Ad-hoc network for *d*=15

4. Conclusion

We described several localized protocols for fast and mostly accurate detection of critical links and nodes in ad hoc networks. Existing algorithms for detection of critical links and nodes have a high complexity and require knowledge of the topology of a network. This paper showed that the localized criteria could successfully be used in order to detect critical links/nodes. Our experiments show that the localized algorithms are reasonably accurate on strongly connected graphs. Errors recorded for localized algorithms come from the ring structures that are created in graph generation. These rings are too wide and more hops are needed in order to correctly detect if a link or node is critical.

In an upcoming companion article, we applied, with suitable modifications and additional criteria, the proposed localized algorithms for detection of critical links and nodes for service replication. In this scenario, a particular route between a client and a server node is monitored for the presence of critical links or nodes. If such node or link is detected, an alternate service in the network is searched, or service is replicated.

The proposed localized partition detection schemes may be applied in sensor network scenarios. Sensors that detect their criticality or existence of critical links may report to monitoring center, asking for deployment of additional sensors in the area, or wakening up some nearby sleeping sensors. We intend to consider this application of localized partition detection for designing sensor activity scheduling protocols.

We have considered unit graph model in our experiments. However, our protocols based on topological knowledge are graph based protocols, and can be applied for ad hoc networks with non-uniform transmission radii. Protocols based on positional information require to know also transmission radii of nodes in local neighbourhood in addition to their positions.

We originally expected to observe much more significant impact of density on the detection ratios. We expected that critical densities for connectivity will also somehow represent significant changes in detection ratios. However, our experimental results did not indicate any relation. This could be due in part to performing experiments only for connected unit graphs. It is certainly important to further study the relation of connectivity probability and detection ratios.

We are currently designing localized algorithms for deciding, at each node, whether or not graph is *k*-connected. Another interesting problems is to assign, in localized manner (as opposed to globalized solution [HIM]), transmission radii to each node so that the network is *k*-connected with high probability.

REFERENCES

[ABS] Duque Anton, M., Bruyaux, F., Semal, P.: Measuring the survivability of a network: connectivity and rest-connectivity; European Transactions on Telecommunications, 2000.

[B] S. Baase, Computer Algorithms, Introduction to Design and Analysis, Addison Wesley, 1988, 184-191.

[BHM] M. Bahramgiri, M. Hajiaghayi, V.S. Mirokni, Fault-tolerant and 3-dimensional distributed topology control algorithms in ad hoc networks, IEEE ICCCN, 2002.

[CYD] G. Cao, L. Yin and C.R. Das, Cooperative cache-based data access in ad hoc networks, IEEE Computer Magazine, Febr. 2004, 32-39.

[DBS] M. Duque-Anton, F. Bruyaux, P. Semal, Measuring the survivability of a network: connectivity and rest-connectivity, European Transaction of Telecomunications, 11, 2, 149-159, 2000.

[GC] D. Goyal and J. Caffery, Partitioning avoidance in mobile ad hoc networks using network survivability concepts, Proc. IEEE Int. Symp. Computers and Communications ISCC, Taormina, Italy, July 2002, 553-558.

[H] T. Hara, Replica allocation methods in ad hoc networks with data update, Mobile Networks and Applications 8, 2003, 343-354.

[HIM] M. T. Hajiaghayi, N. Immorlica, V.S. Mirrokni, Power optimization in fault-tolerant topology control algorithms for wireless multi-hop networks, Proc. ACM MOBICOM, Sept. 2003.

[HSC] M. Hauspie, D. Simplot, and J. Carle. Partition detection in mobile ad-hoc networks. *Proc. 2nd IFIP Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET 2003)*, (Mahdia, Tunisia, 2003).

[JS] L. Jia and C. Scheideler, On local algorithms for topology control and routing in ad hoc networks, *Proceedings of the 15th Annual ACM Symposium on Parallel Algorithms and Architectures,* pages 220-229, June 2003.

[K] H. Koskinen, Statistical model describing connectivity in ad hoc networks, Proc. Modeling and optimization in mobile, ad hoc, wireless networks WiOpt, INRIA, Sophia-Antipolis, March 2003.

[KMP] G. Karumanchi, S. Muralidharan, R. Prakash, Information dissemination in partitionable mobile ad hoc networks, Proc. IEEE Symp. Reliable Distributed Systems, October 1999.

[LR] Q. Li and D. Rus, Communication in disconnected ad hoc networks using message relays, ACM MOBICOM 2000; J. Parallel and Distributed Computing, 63, 2003, 75-86.

[M] U. Manber, Introduction to Algorithms, A Creative Approach, Addison Wesley, 1989, p. 217-225.

[PC] V.D. Park and M. S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. *IEEE INFOCOM '97*, Kobe, Japan, April 1997.

[SCN] S.H. Shah, K. Chen, and K. Nahrstedt. Cross-layer design for data accessibility in mobile ad hocnetworks. In *Proc. of 5th World multiconference on systemics, cybernetics and informatics (SCI 2001), Orlando, Florida*, July 2001; Wireless Personal Communications, vol. 21, pp. 49-76, 2002.

[SRV] I. Stojmenovic, M. Russell, and B. Vukojevic, Depth first search and location based localized routing and QoS routing in wireless networks, Computer Science, SITE, University of Ottawa, TR-00-01, January 2000; Computers and Informatics, 21, 2, 2002, 149-165.

[T] R. Tarjan, Depth first search and linear graph algorithms, SIAM J. Computing, 1, 2, 146-160, 1972.

[VB] A. Vahadat, D. Becker, Epidemic routing for partially connected ad hoc networks, Technical Report CS-200006, Duke University, April 2000.

[WL1] K.H. Wang, B. Li, Efficient and guaranteed service coverage in partitionable mobile ad hoc networks, INFOCOM, 2002.

[WL2] K.H. Wang and B. Li, Group mobility and partition prediction in wireless ad hoc networks, IEEE ICC, April 2002.