Acta Universitatis Sapientiae
Electrical and Mechanical Engineering, 4 (2012) 5-32

Cluster Formation in Wireless Mesh Networks

Dezső GÁBOS, Mihály VARGA

Communications Department,
Faculty of Electronics, Telecommunications and Information Technology,
Technical University of Cluj-Napoca,
e-mail: dezsogabos@gmail.com, Mihaly.Varga@com.utcluj.ro

Manuscript received November 1, 2013; revised January 13, 2014.

Abstract: Cluster formation in wireless mesh networks simplifies the maintenance of the entire network and allows the usage of decentralized routing algorithms. The defining factors in choosing the routing algorithm is the type of the network (MANET or WSN) and the node density. This paper consists of the description of a simulation platform for cluster formation in cooperative networks, which permits analysis and evaluation of different routing algorithms in such networks. The generic cluster formation algorithm was implemented in such a way, that each node has information only on neighbouring nodes. A couple of simple algorithms are presented which may be applied to the generated clusters (routing and data processing) to demonstrate the simulation capabilities offered by this platform. These algorithms include: maximum flow minimum cut of Ford Fulkerson combined with Dijkstra's routing algorithm.

Keywords: Ford Fulkerson, graph theory, wireless mesh networks

1. Introduction

Communications are evolving more and more in the direction of ad-hoc networks. The reason why these networks are gaining ground is because they do not require a well-defined infrastructure (base stations, routers, cables) and can be reconfigured at any time, adding or eliminating terminals or nodes (sensors, user terminals). A wireless ad-hoc network consists of a collection of autonomous nodes or terminals which communicate among themselves and form multi-hop radio networks maintaining connection in a decentralized manner. These types of networks are commonly used in areas where the fixed infrastructure has been destroyed or it is missing [1]. For example, a class of

students has to interact during a lecture, friends or business partners meet at an airport and need to exchange files, or an intervention team has to create connection with the outside of the affected area. In these situations a collection of mobile hosts with wireless network interfaces can create a provisory network without an existing infrastructure or centralized administration. This type of network is called an ad-hoc network [2].

Ad-hoc networks are based on ideas like reconfigurability, self-organization, dynamics, omnipresence and most importantly are centred around the needs of the user [3]. These are the main characteristics of next generation communications. An amazing development has taken place in the domain of wireless sensor networks, personal networks, wireless mesh networks, multi-hop communications and their integration in cellular systems. A multi-hop, multi-cluster system requires the capability to dynamically adapt to constant changes in the network. With the cluster formation algorithm the mobile nodes are divided in groups called clusters. The nodes in a certain cluster can be a gateway, simple nodes or the root of the cluster (called cluster head). Cluster heads (CH) are responsible for coordination of operations in the cluster, forming of new clusters and topology maintenance. Furthermore they have the responsibility of parting resources between members of a cluster.

In these types of networks the reconfiguration of the cluster heads is a necessity. The algorithm should not change the cluster configuration frequently. The most important advantages of these systems are: the scattered usage of applications, the possibility of routing optimization, efficient treatment of mobility, a better usage of bandwidth, aggregation of topology based information and minimization of necessary storage for information [3].

In a wireless ad-hoc network each node has the capability of forwarding messages from other nodes in the network. The route to the destination is determined by the connectivity of the whole system. The main characteristics of an ad-hoc network are: minimal configuration requirements and fast implementation. To maintain connections between nodes, each and every node has to be a router in itself. Each node functions both as a host and a router, and the control over the topology of the system are distributed between them. The topology of the network becomes dynamic because the connections between the nodes vary due to the addition and elimination of nodes in the system and due to their mobile nature. This is why the necessity of new routing protocols appeared. The two main types of ad-hoc networks are Mobile ad-hoc networks (MANET) [4] and wireless sensor networks (WSN) [5].

These systems need to possess the capability of self-organization on the physical layer. In the absence of a fixed infrastructure, a fully functional network can be achieved only through the possibility of cooperation between the nodes [6]. The topology changes are random, the connections between the

nodes have variable available bandwidths, so the need for decentralized routing algorithms appeared [7]. The discovery of the new topology and the delivery of data packets becomes the nodes' responsibility. In fixed networks the main algorithm is based on the cost function (shortest path), but this cannot be implemented in ad-hoc networks. The main problems are: free space pathloss and fading, interference, power consumption, topology changes [8]. The network is able to counter these effects by adaptive routing [9].

Each node or mobile device is equipped with a transmitter and a receiver, so without a set of rules the topology becomes arbitrary. A packet can arrive to the destination directly or through other intermediary nodes, which forward the packet. The decisions when building the topology and self-discovery mechanisms are key elements for a well-functioning ad-hoc network. The optimization and standardization cannot be restricted to a single layer [10] [11].

This paper approaches the problem of routing and cluster formation in wireless ad-hoc networks, by studying hierarchical structures, and well known cluster formations. Cluster formation is a method for constructing and maintaining hierarchical structures in a mobile ad-hoc network. The main parameters are: connectivity, stability of connections, bandwidth (quality of the connection). In a network based on clusters, the nodes can form any hierarchical structure. The most important advantage of an approach based on cluster formation is the improvement of routing efficiency, scalability and the minimization of power consumption. The dynamic forming of clusters help reduce the complexity of the nodes, packet length and help the multi-path routing. Because the size of an ad-hoc network is unlimited, partitioning it in clusters makes it manageable.

2. Routing and cluster formation in wireless ad-hoc networks

Cluster forming transforms a physical network in a virtual network, which contains interconnected clusters, or groups of mobile nodes [12]. Even with the nodes being identical in their capabilities, some nodes can be chosen to form the backbone of the network. These nodes become gateways and *cluster heads* (CH). The cluster heads are the nodes which are responsible of routing the messages to all the nodes in their cluster [13]. The gateway nodes are the nodes at the extremities of the cluster, and usually communicate with the gateway nodes from other clusters. The backbone of the wireless network can be used for routing packets, broadcasting routing packets or both in the same time. Due to the mobility of the nodes this backbone needs the capability of being reconstructed fast, as the nodes enter or leave the coverage area of the CH.

Routing protocols are divided into two categories based on how and when routes are discovered, but both find the shortest path to the destination.

Proactive routing protocols are table-driven protocols, they always maintain current up-to-date routing information by sending control messages periodically between the hosts which update their routing tables. When there are changes in the structure then the updates are propagated throughout the network. The proactive routing protocols use link-state routing algorithms which frequently flood the link information about its neighbours.

Other routing protocols are on-demand routing protocols, in other words reactive ones, which create routes when they are needed by the source host and these routes are maintained while they are needed. Such protocols use distance-vector routing algorithms [14], they have vectors containing information about the cost and the path to the destination. When nodes exchange vectors of information, each host modifies its own routing information when needed. The ad hoc routing protocols are usually classified as a pure proactive or a pure reactive protocol, but there are also hybrid protocols. This only concerns flat routing protocols, but there are also hierarchical and graphic position assisted routing protocols [15].

The main routing and cluster formation algorithm presented on literature are: CBLARHM (Cluster Based Location-Aware Routing Protocol for Large Scale Heterogeneous Mobile Ad-Hoc Networks), CBMPR (Cluster Based Multipoint Relay Protocol), CRAM (Cluster head based Routing Algorithm), SCRAM (Scenario Based Routing Algorithm), LEACH (Low Energy Adaptive Clustering Hierarchy), HEED (Hybrid, energy efficient, distributed clustering), PEGASIS (Power efficient gathering in sensor information systems), ERA (Energy Residue Aware), PEBECS (Partition Energy Balanced and Efficient Clustering Scheme), RCSDN (Distributed Balanced Routing Algorithm with Optimized Distribution), and many more.

There are two important mechanisms used in the *Cluster Based Location-Aware Routing Protocol for Large Scale Heterogeneous Mobile Ad-Hoc Network* (CBLARHM) [16] to improve the MANET performance. Node clustering is an efficient technique to mitigate the topology changes in MANET. It stabilizes the end-to-end communication paths and improves the networks scalability so that the routing overhead does not become tremendous in large scale MANET. Another key challenge is controlling the total number of nodes involved in a routing establishment process so as to reduce the total routing overhead of the MANET. The mechanism used is to use the geographical location information provided by global positioning systems (GPS) to assist in routing. Instead of searching the route in the entire network blindly, position-based routing protocol uses the location information of mobile nodes to confine the route searching space into a smaller estimated range. Simulation results have shown that CBLARHM outperforms other protocols significantly in route setup time, routing overhead and collision, and simultaneously maintains a low

average end-to-end delay, as well as low route discovery frequency [16].

The *Optimized Link State Routing* (OLSR) [17] protocol is a proactive routing protocol. The key concept used in the protocol is that of multipoint relays (MPRs). MPRs refer to selected routers that can forward broadcast messages during the flooding process. The idea of multipoint relays is to minimize the flooding of broadcast packets in the network by reducing duplicate retransmissions in the same region. Although OLSR provides a path from source to destination, it is not necessarily the shortest path, because every route involves forwarding through a MPR node. A further disadvantage is that OLSR also has routing delays and bandwidth overhead at the MPR nodes as they act as localized forwarding routers.

The Dynamic Source Routing (DSR) [15] protocol is a reactive routing protocol for MANETs. The key feature of DSR is the use of source routing, which means the sender knows the complete hop-by-hop route to the destination. A complete list of intermediate nodes to the destination kept in the header of each data packet. Scalability and poor performance in high mobility and heavy traffic loads are disadvantage of DSR, because DSR relies on blind broadcasts to discover routes. AODV [17] is essentially a combination of both DSR and DSDV. It borrows the basic on-demand mechanism of Route Discovery and Route Maintenance from DSR, plus the use of hop-by-hop routing, sequence numbers, and periodic beacons from DSDV. AODV is loop-free due to the destination sequence numbers associated with routes. It creates routes only on-demand, which greatly reduces the periodic control message overhead associated with proactive routing protocols. Similar to DSR, poor scalability is a disadvantage of AODV. Bypass-AODV, an extension of AODV, uses a specific strategy of cross-layer MAC-interaction to identify mobility-related link breaks, and then setup a bypass between the broken link end nodes via an alternative node. [18]

Low Energy Adaptive Clustering Hierarchy (LEACH) [19] is a hierarchical protocol in which most nodes transmit to cluster heads, and the cluster heads aggregate and compress the data and forward it to the base station(sink). Each node uses a stochastic algorithm at each round to determine whether it will become a cluster head in this round. LEACH assumes that each node has a radio powerful enough to directly reach the base station or the nearest cluster head, but that using this radio at full power all the time would waste energy.

Nodes that have been cluster heads cannot become cluster heads again for P rounds, where P is the desired percentage of cluster heads. Thereafter, each node has a $1/P$ probability of becoming a cluster head in each round. At the end of each round, each node that is not a cluster head selects the closest cluster head and joins that cluster. The cluster head then creates a schedule for each node in its cluster to transmit its data.

All nodes that are not cluster heads only communicate with the cluster head in a TDMA fashion, according to the schedule created by the cluster head. They do so using the minimum energy needed to reach the cluster head, and only need to keep their radios on during their time slot.

Power efficient gathering in sensor information systems PEGASIS [21]- sensor webs consisting of nodes with limited battery power and wireless communications are deployed to collect useful information from the field. Gathering sensed information in an energy efficient manner is critical to operate the sensor network for a long period of time. In a data collection problem is defined where, in a round of communication, each sensor node has a packet to be sent to the distant base station. If each node transmits its sensed data directly to the base station then it will deplete its power quickly. The LEACH protocol presented in is an elegant solution where clusters are formed to fuse data before transmitting to the base station. By randomizing the cluster heads chosen to transmit to the base station, LEACH achieves a factor of 8 improvement compared to direct transmissions, as measured in terms of when nodes die. In PEGASIS, each node communicates only with a close neighbour and takes turns transmitting to the base station, thus reducing the amount of energy spent per round. Simulation results show that PEGASIS performs better than LEACH by about 100 to 300% when 1%, 20%, 50%, and 100% of nodes die for different network sizes and topologies [21].

These algorithms are mostly about choosing a cluster head (CH), which take care of the routing between the clusters. Most of them concentrate on the power management, and try to find a method to minimize the overhead, to reduce the power consumption, to prolong the life of a node, to extend the system duration by finding a balance in the node usage. In the case studied by us, we do not need these algorithms, because we assume that the nodes are connected to a power source, and power consumption is not a problem.

In this paper we consider that a cluster consist from the set of nodes which can communicate together without using an external communication channel, i.e. there exist at least one route between any two elements of the cluster with capacity higher than zero. The problem is finding a route, on which we can transmit with the highest transfer rate. We can find this, by applying the Maximum spanning tree algorithm, because our edges are defined by transfer rates. The novelty brought to this algorithm is the method of solving this graph theory problem in a decentralized way. Each and every node makes decisions based on the information known to it, and has no access to information about the whole system.

A. Ford Fulkerson algorithm

A commonly met question in networks is: "How much is the maximum data transfer rate between two nodes?" A good example would be the maximum number of phone calls between two cities, or the maximum number of cars that can travel between two cities. An infinite transfer rate is impossible, because the telephone lines or roads have a limited capacity. On the other hand, there are more routes on which you can arrive at your destination. Finding the maximum flow between two points means the analysis of all the routes between these two nodes [22]. When the system is a network, the edges represent the channels with limited capacity. To find the maximum flow, each edge needs to be used at maximum capacity. Another possible problem can be the asymmetry of these edges (difference in capacity between directions). In the following figures, it will illustrate the principle of this algorithm, step by step.

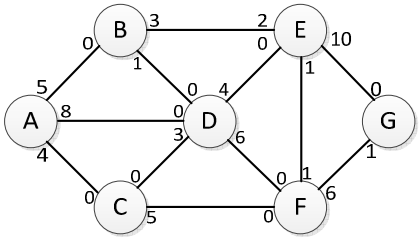

Figure 1: Undirected graph with asymmetric capacities.

Between points A and G there is a transfer rate of 4 on the A-D-E-G route, a transfer rate of 3 on the A-B-E-G route and a transfer rate of 4 on the A-C-F-G route. The bottleneck is the edge with the smallest transfer rate on a given route. This set of routes sum up to a total transfer rate of 11 from A to G, but this is not the maximum flow. There are unused edges between the two nodes. We need a better algorithm, to calculate the maximum flow, and we will implement it with the Ford Fulkerson method. The basic idea of this algorithm is the manipulation of the assigned data flows to the different routes from the source node to the destination. The main steps of the algorithm are:

1. Finding any of the possible routes from the source node to the destination, with a positive flow. If there are no more routes, the algorithm ends.
2. The determination of f, the maximum flow along this route, which equals the smallest edge capacity from the routes edges.
3. Decreasing all edge capacity values from the routes' edges with the value of *f*, on the direct route and adding it to the edge capacities on the inverse route.
4. Repeating from step 1.

At the end of the algorithm, the sum of the flows along the routes found at step 1 gives us the maximum flow between the source and destination.

B. Step by step implementation of the Ford Fulkerson algorithm

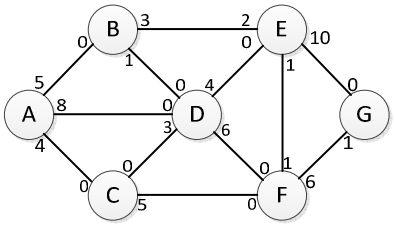

Figure 2: Original Graph.

The original graph contains nodes and edges; and we define a capacity on each edge. These capacities do not have to be symmetric. A good example would be a road between two cities. The number of car strips in one direction must not always be equal to the car strips in the other direction. This principle can also be applied with radio transmissions. The channel's characteristics in point A is not identical with the channel characteristics from point B. This way, we can define different capacities on the same edge in different directions.

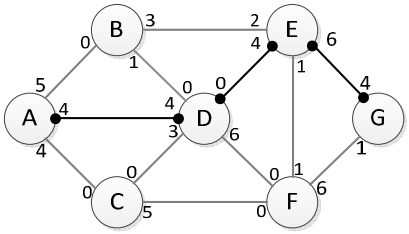

Figure 3: Graph illustrating the first route.

We chose the A-D-E-G route. The bottleneck will be the D-E edge, which has the capacity of 4. This capacity is subtracted from all edges on the direct route between A and G, and it is added to the inverse route. This way the capacity between A-D becomes 4 from 8, on D-E it becomes 0 from 4, and on E-G it becomes 6 from 10. On the inverse route, D-A becomes 4 from 0, E-D becomes 4 from 0 and G-E becomes 4 from 0.

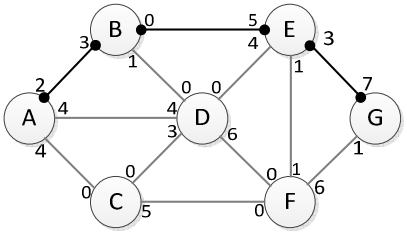

Figure 4: Graph illustrating the second route.

We chose the A-B-E-G route. The bottleneck is the B-E edge, with the capacity of 3. This capacity is subtracted on the direct route, so the A-B capacity becomes 2 from 5, the B-E capacity becomes 0 from 3 and the capacity of E-G edge becomes 3 from 6. On the inverse route G-E becomes 7 from 4, E-B becomes 5 from 2 and B-A becomes 3 from 0.

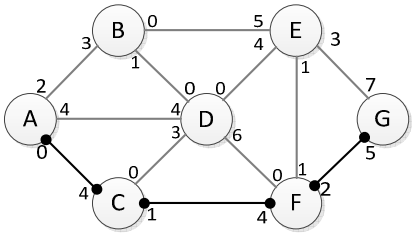

Figure 5: Graph illustrating the third route.

We chose the A-C-F-G route. The bottleneck is the A-C edge with the capacity of 4. On the direct route the A-C capacity becomes 0 from 4, the C-F capacity becomes 1 from 5, and between F-G the capacity becomes 2 from 6. On the inverse route C-A becomes 4 from 0, F-C becomes 4 from 0 and G-F becomes 5 from 1.

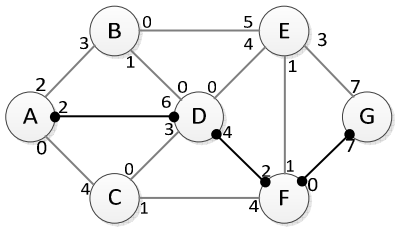

Figure 6: Graph illustrating the fourth route.

We chose the A-D-F-G path. The bottleneck is the F-G edge, with the capacity of 2. On the direct route A-D becomes 2 from 4, D-F becomes 4 from 6, and F-G becomes 0 from 2. On the inverse route the capacity of G-F becomes 7 from 5, F-D becomes 2 from 0 and D-A becomes 6 from 4.

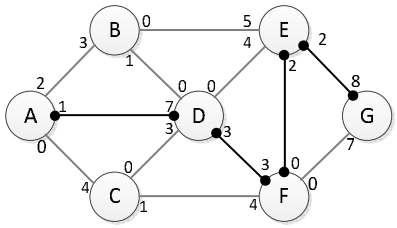

Figure 7: Graph illustrating the fifth route.

We chose the A-D-F-E-G route. The bottleneck is the F-E edge with the capacity of 1. On the direct route, A-D becomes 1 from 2, D-F becomes 3 from 4, F-E becomes 0 from 1, E-G becomes 2 from 3. On the inverse route G-E becomes 8 from 7, E-F becomes 2 from 1, F-D becomes 3 from 2, D-A becomes 7 from 6.

As we can see, we have no positive flows left between nodes A and G. The algorithm ends. After finishing the algorithm, the maximum flow between node A and node G will be the sum of the 5 flows on the 5 different routes. This way, the maximum flow becomes 4+3+4+2+1=14.

The question is, how much flow should there be on each edge, and in what direction? We find the answer looking at the difference between the initial capacity and final capacity. A positive difference shows us a flow in association with the direct route, a negative difference is ignored. We can observe the principle of flow conservation at a node. For example, the flows entering in node F equal the sum of the flows exiting node F, and equal to 7.

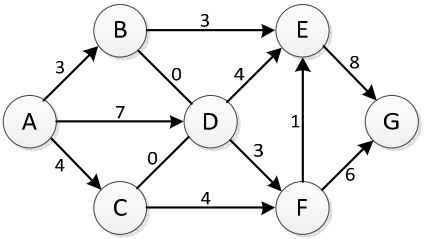

Figure 8: Final fluxes.

As we can see, it becomes harder and harder to find a positive flow, as the algorithm progresses, and the easily identifiable routes disappear.

This in fact is not a problem, because the next route can be found with algorithms like depth-first and breadth first. The remaining question is: why do we need to change the edge capacities on the inverse route? The answer is simple: the edge capacities on the inverse route can be seen as a convention for indicating a restorable flows if it is needed.

The problem of maximum flow is in direct association with the problem of minimum cut. A cut is a set of directional edges, which contains at least one edge from all routes between the source node and destination node. In other words, if the edges are eliminated, the flow from the source node to the destination node is completely cut off. The value of the cut is the sum of the edge capacities on the source-destination direction on the edges present in the cut. The problem is finding the cut in a way, that it has the smallest value from all possible values. An inefficient method would be finding the values of all cuts, and choosing the one with the smallest value. The number of these cuts is very large, so it is almost impossible to find all possibilities in a larger network,

even with nodes with a high computational power. A better idea is using the max-flow/min-cut theorem: for any network having only one source and one destination, the maximum flow from the source to the destination equals the value of minimum cut for all the cuts in the network.

A lot of applications use this theorem, including the search for congested nodes in traffic, in telecommunication networks, production lines and so on. Other algorithms with more complex implementations and better efficiency are the Goldberg algorithm and Edmons-Karp algorithm.

3. Simulation platform

The flow chart of the implemented simulation platform is presented in Fig. 9.

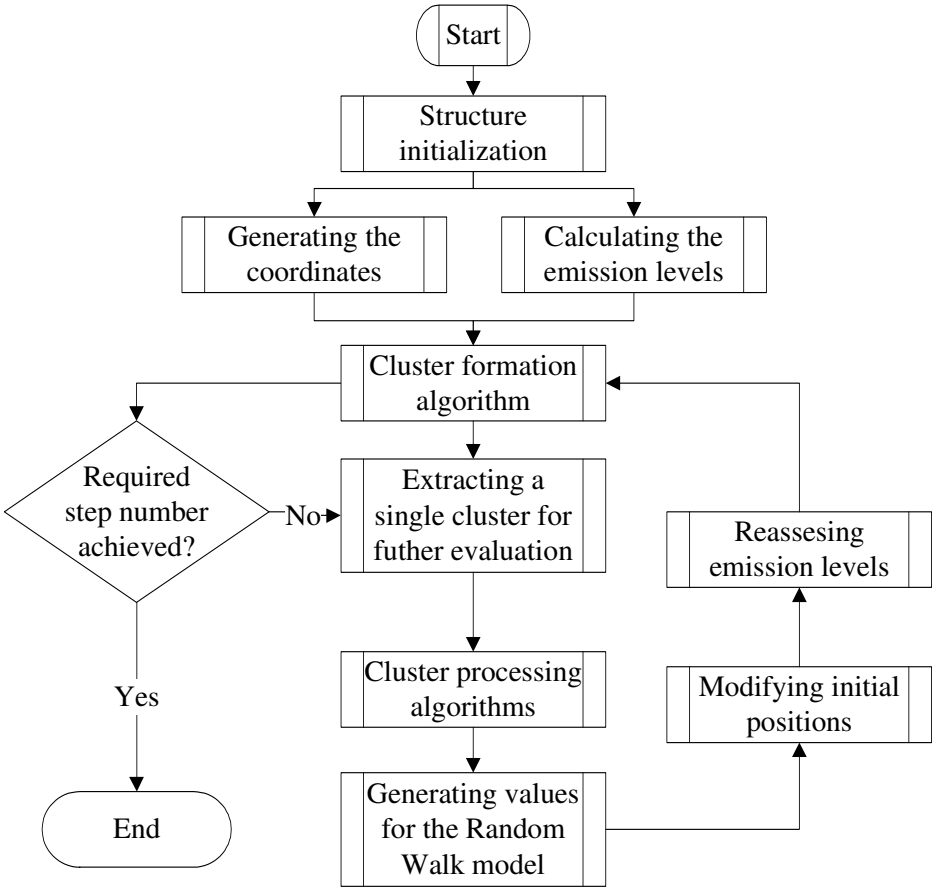

Figure 9: Flow chart for the simulation platform implementation.

The initialization of structures are based on an algorithm which generates random coordinates (x and y separately), and saves them in a structure with an equal length with the number of users. The coordinates are generated in accordance with the surface on which we plan to implement the algorithm. After calculating the necessary power level, based on the distance (so that we can still use a modulation at the edge of the coverage area), we apply the cluster formation algorithm. These algorithms do not take in consideration the frequency usage, and minimization of interference, but based on node density and number of nodes form groups which are delimited only by their own power. For example a node can be from a corner of the network, and another one from the other corner, and between the two nodes we can find other clusters, which make it possible for our two nodes to communicate.

After the clusters are formed, we delimit a single cluster, on which we apply our algorithms, to evaluate its behaviour and characteristics. In our case we chose the Maximum Spanning Tree algorithm, and an improved implementation of Ford Fulkersons algorithm for maximum cut/minimum flow. These algorithms can be anything, including routing algorithms (as demonstration, the Dijkstra's algorithm is also implemented). When we are finished working with the cluster, we can proceed to the next cluster, or we can work with more clusters in parallel. We chose to work with only one cluster, as it is sufficient to demonstrate the capabilities of this simulation platform, and visualize these processes by creating animations, which illustrate the algorithm step-by-step.

After this, we generate a random value for speed and a random value for direction. These two values are defined between a maximum and minimum value, so that the node speed would not influence the transmission rates. The speeds are relatively small (a Random Walk model is used, so the influence of Doppler effect can be neglected). As we generate these values, we transform from polar coordinates to Cartesian coordinates, and add the values to the initial position of the nodes. We apply algorithms to find the neighbouring nodes, and delimit the new clusters. They can be totally different from the initial clusters, because a node which belonged initially to another cluster gets close to a second clusters, and the node becomes a sort of point of connection, which makes the communication between the two clusters possible. There are nodes which do not belong to any clusters, due to the distance from other clusters, or the small power of emission.

After each cycle we save the data which will be processed, and an image which will become a frame in the final animation. There are two final animations, one to illustrate the random walk algorithm together with the cluster formation, and an animation which illustrates the intermediary processing of clusters, in this case, the Ford Fulkerson algorithm

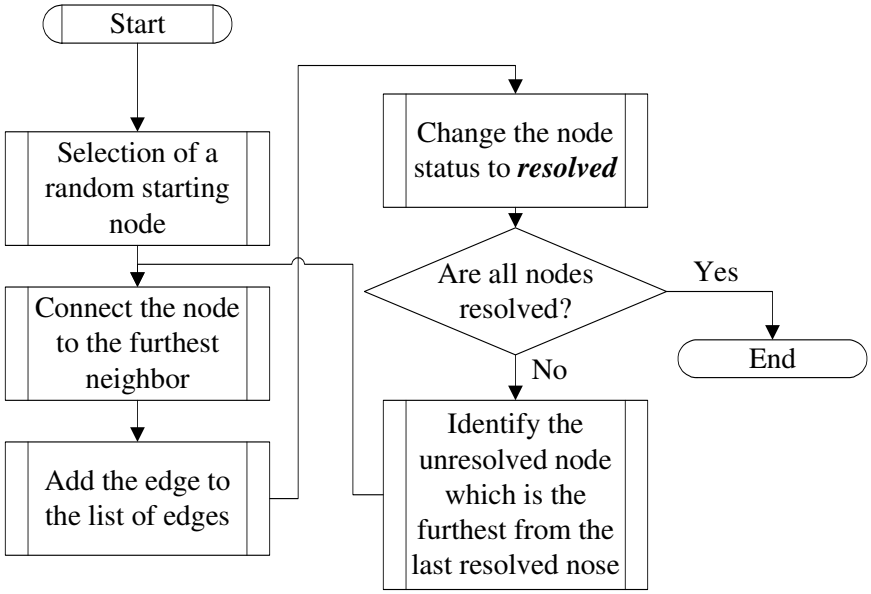

Figure 10: Flow-chart for maximum spanning tree

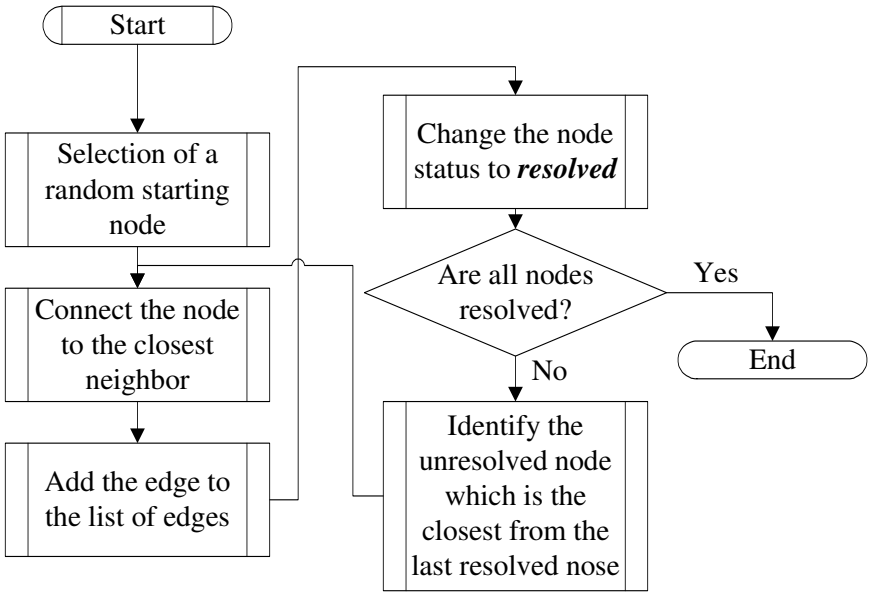

Figure 11: Flow-chart for minimum spanning tree.

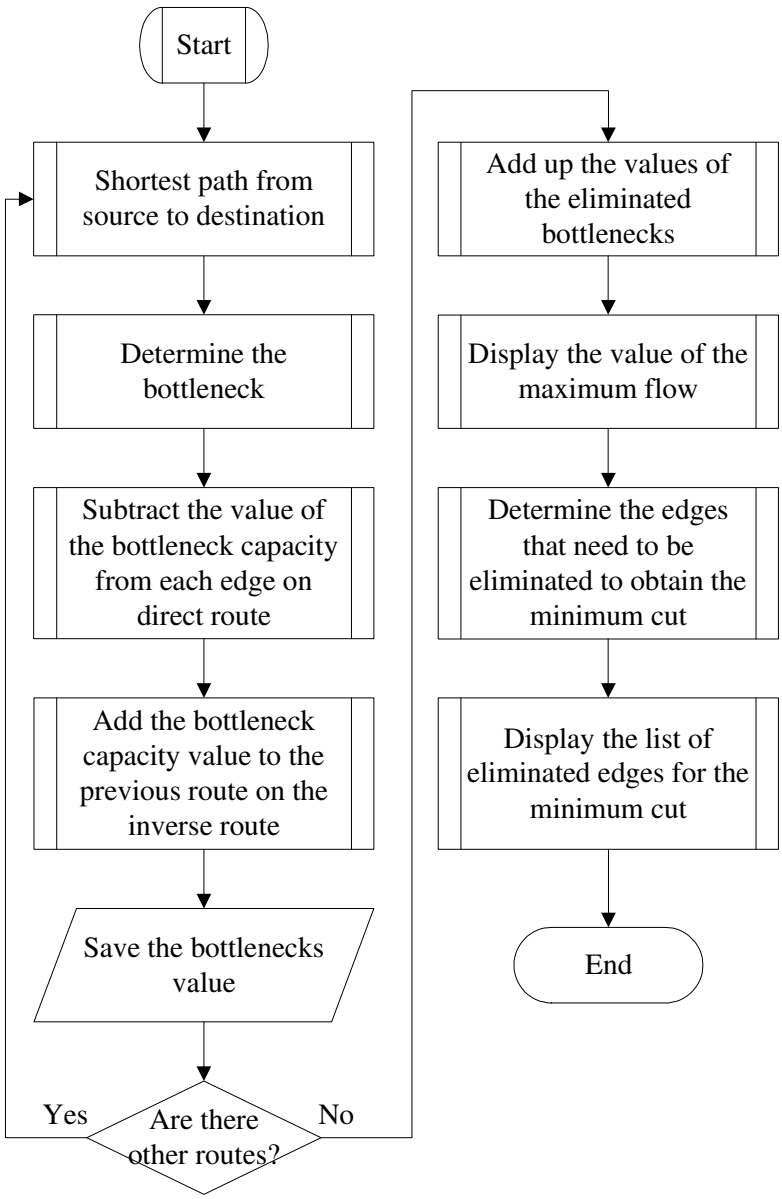

Figure 12: The improved Ford-Fulkerson algorithm.

After each cycle we save the data which will be processed, and an image which will become a frame in the final animation. There are two final animations, one to illustrate the random walk algorithm together with the cluster information, and an animation which illustrates the intermediary processing of

clusters, in this case, the Ford Fulkerson algorithm. This algorithm usually uses depth-first and breadth first algorithms. Here it is improved by applying Dijkstra's algorithm [20] for the shortest path. When routes are eliminated, the routing table is updated, and there is no need to reapply a depth first algorithm. The cluster information algorithm is a personal implementation. It uses the information saved in structures associated with each node in particular. The basic structure is the following:

```
vecin=struct('vec_nr',{},'vec_list',{},'head_list',{}
,'pwr_list',{});
util=struct('x',{},'y',{},'vecin',{},'ni',{},'clust_num',{}
,'flag',{});
clusters=struct('node',{},'elements',{});
```

Based on these informations, we can group nodes in clusters. The structure will contain the *x* and the *y* coordinates of the node, the *vecin* structure contains a list with the nodes neighbours, *ni* is a helping index, *clust_num* is the index of the cluster to which the current node belongs, and *flag* is a Boolean variable, helping in the algorithm. The structure *vecin* contains the number of neighbours, a list of the neighbours, a list with roots when the clusters are formed and a list of maximum transfer rates on all the edges associated with the current node. These maximum transfer rates were defined by the Euclidean distances between the nodes. The distance is calculated between the node and all of his neighbours, and from the distance the free space path loss is determined. This attenuation is subtracted from the emission power of the source node. From this SNR value we can determine with the help of the Hartley Shannon theorem the maximum transfer rate that can be achieved over this channel. The maximum transfer rates are used only in the cluster processing algorithms, and not in the cluster formation algorithms. The cluster formation algorithms are based only on distances and emission powers.

Fig. 13 illustrates the complexity of the structures we work with. Each element from the structure contains a list of Neighbour structures. This allows us to the dynamic change of the implementation: with the change of a single variable, all the structures change, and the necessary data is updated, so there are no contradictory information in the tables and structures.

When we work with data associated to a cluster, we can easily extract all the needed information. To do this, we simply use the indexing and the cluster number. When we extract the information from the structures, an adiecency matrix of the extracted cluster is created.

The structure Clusters will be containing all the formed clusters, and each element of the structure contains the number of nodes, and their indexes. This structure is created to exemplify the data extraction process [23]. From the

initial structure we can extract any variable and any structure. This information can be used for a different number of applications and simulations.

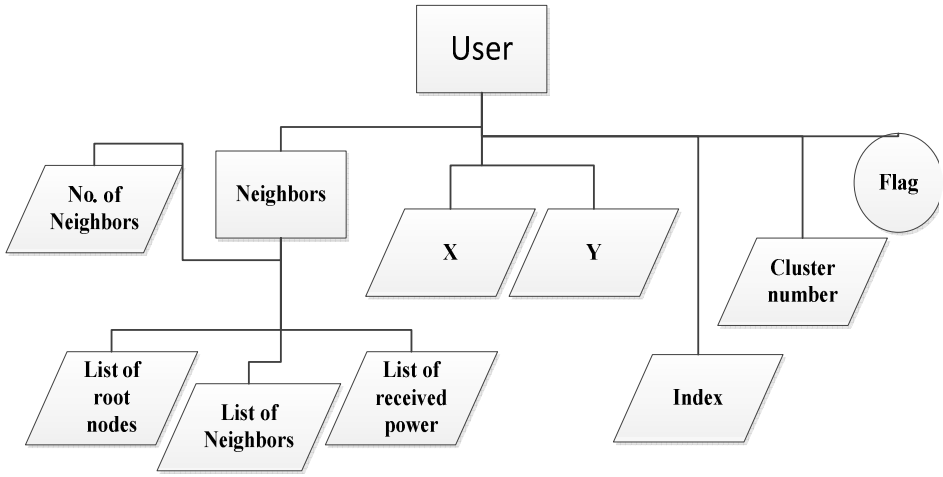

Figure 13: Data structure associated with one node.

4. Numerical results

We decided to compare the band of 2.4 GHz with the band of 5.25 GHz from the point of view of attenuation. (802.11a and 802.11b)[24]. Path gain is the inverse of the free space path loss function (*Fig. 14*).

We decided to use the 2.4 GHz band in this simulation. MANET-s uses both frequencies, so in our implementation can be changed at any time. In the case of WSN the used frequencies are lower, because their power consumption needs to be reduced to prolong the life of the nodes. Today, the frequencies used in wireless sensor networks include 315 MHz, 433 MHz, 868 MHz (in Europe) and 915 MHz in North America, and 2.45 GHz in the medical industry. This leads to the coverage of larger areas with the same number of mobile nodes, but meanwhile leads to greater interferences, due to the usage of the same bands by other applications. The interference signals can be analogical TV signals in the band of 400 MHz, GSM signals in the band of 900 MHz, Bluetooth signals from the GHz domain and interference from other electrical devices such as microwave oven, electro motors, etc. At small frequencies the free space path loss is not the most significant parameter. However, small frequencies require larger antennas. If the node density would allow distances of a couple of meters, and the users would not have antenna size constraints, a low frequency

implementation is the best solution. Frequencies of 70 MHz are used in some cases of wireless sensor networks.

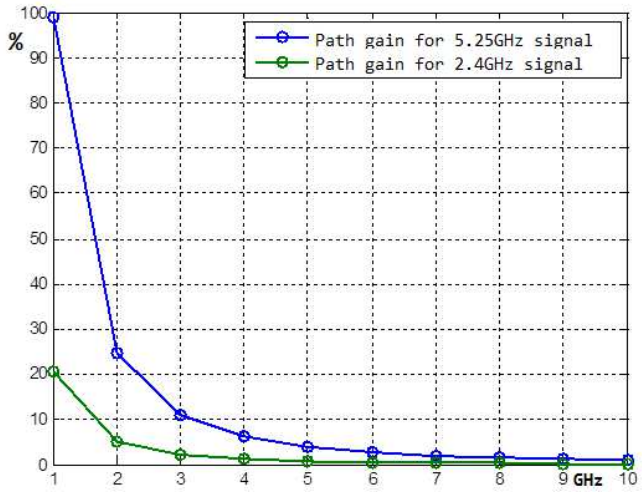

Figure 14: Free space path gain

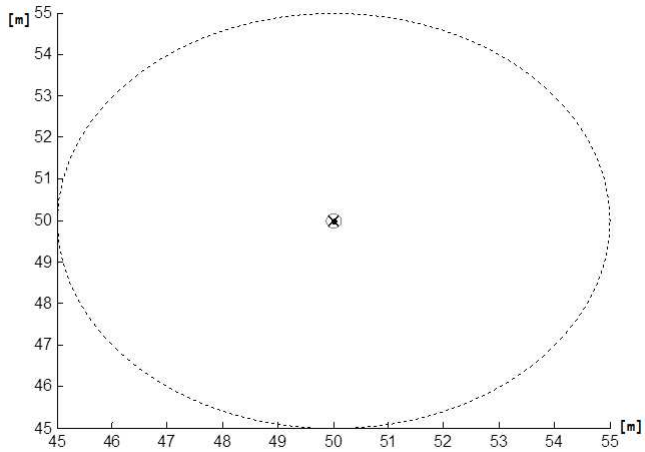

Figure 15: The coverage area of a node

We decided to set the emission power of a node in such a way, that at the border of the coverage area we can still use a modulation with a minimum number of phases (2PSK). This modulation requires a signal noise ratio of 9.5dB. On a distance of 100 m the expression of the free space pathloss is:

$$FSPL = 10\log_{10}\left(\frac{3\cdot 10^8}{4\pi\cdot 100\cdot 2.4\cdot 10^9}\right) \quad (1)$$

The result is 16dB. So we set the power of emission to 9.5dB+16dB = 25.5dB. The coordinates of the users are randomly generated on a finite surface with a Gaussian distribution.

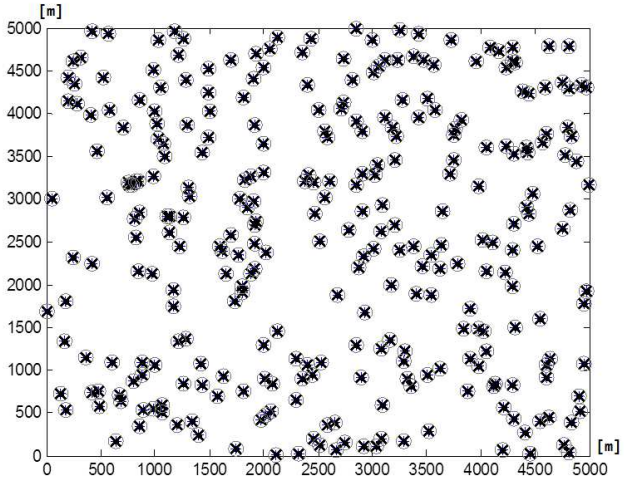

Figure 16: Surface with distributed nodes

We generated the x and y coordinates separately, and created structures containing all the necessary information related to the node. Initially they contain only the node coordinates. The chosen surface is a surface of 5000×5000 square meters and the user/node number is 300.

After establishing the range of all the nodes, we applied an algorithm which calculates all the Euclidian distances to all the neighbouring nodes delimited by this range.

$$d = \sqrt{(x - x_0)^2 + (y - y_0)^2} \quad (2)$$

So to the initial structure was completed with a list of neighbours containing the index of the neighbouring nodes, so we can obtain their coordinates at any given time. This was necessary, because in a system with nodes, there is not even one node which has knowledge about the whole network, they only communicate directly with neighbouring nodes. We decided that the cost function associated to the edges between the nodes to not be the Euclidian distance, but maximum transfer rates. As long as we have information about the

neighbours, we can apply the Hartley-Shannon theorem, which defines the maximum possible transfer rate on a channel with noise.

$$C = B\log_2\left(1+\frac{S}{N}\right) \tag{3}$$

We consider that B is 1 Hz (we evaluate spectral efficiency), and the signal noise ratio is defined by the difference between the emission power level of the node and free space path loss between the node and his neighbour, in other words the effective distance between the two nodes.

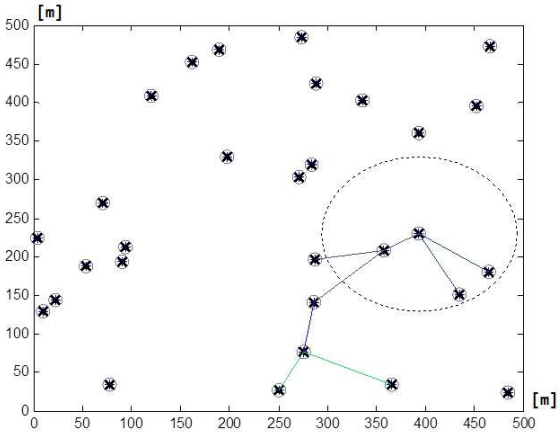

Figure 17: Determined grades of vicinity.

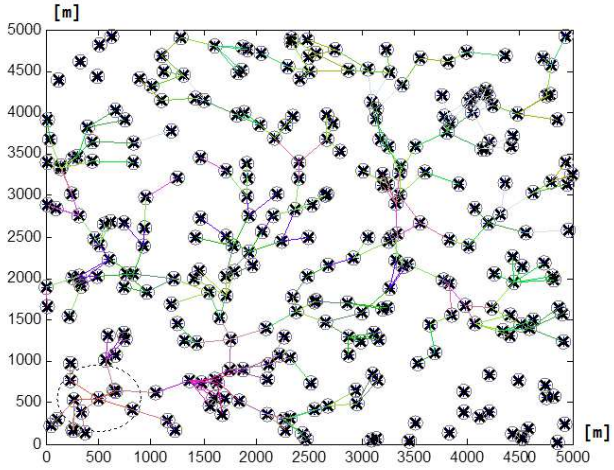

Figure 18: Generating the vicinity level map with a high user density.

If we want a walk on the graph, if we start from the information stored in one node, we can arrive to all the nodes calculated as neighbours, and defined as 1st grade neighbours. From them we can arrive at the neighbours of the neighbours, which are 2nd grade neighbours to the initial node. With this algorithm we can form a spanning tree, which contains information about the maximum transfer rates on the edges. When the node density is too small, the graph will not include all the nodes, but form a cluster instead with a small number of nodes. The users inside a cluster can communicate, and all that we have to do is to apply an inter-cluster routing algorithm, to connect all the formed clusters

We decided to apply the algorithm deciding the vicinity levels to a network with larger user density, and observed that by defining a power level above a certain limit, the whole network will be interconnected. We chose a random source node and a random destination, and illustrated this vecinity level map with different colours.

The figures illustrate that in these networks, where the node density is high, communication can be accomplished between any two nodes, moreover there is a large number of possible routes between them. We decided to demonstrate with the Ford-Fulkerson algorithm, that these transfer rates are limited because of the bottlenecks, which appear due to the large distances. The simplest way to put this is: on a route from node A to node B the maximum transfer rate is limited by the minimum transfer rate on one of the edges on this route.

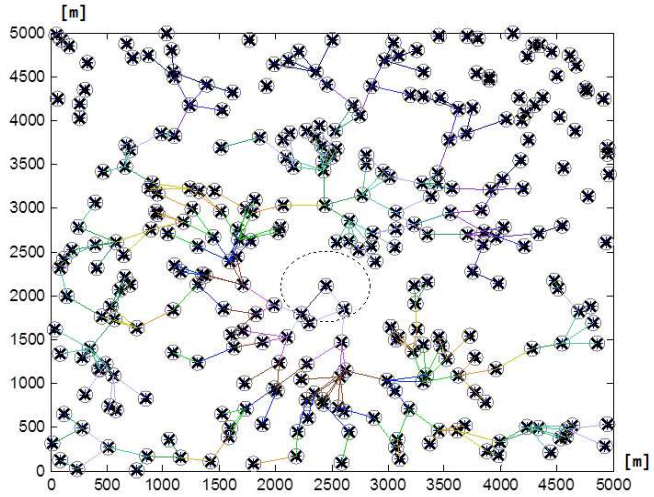

Figure 19: Generating the vicinity level map with a high user density with a random source and random destination node.

As a next step, we reduced the density of the nodes by reducing the total number of nodes on the defined surface. We can arrive at the same results, if we reduce the emission power of the nodes. As we can clearly see, not all nodes are connected to a graph. We applied an algorithm which groups the nodes which can communicate with each other into clusters.

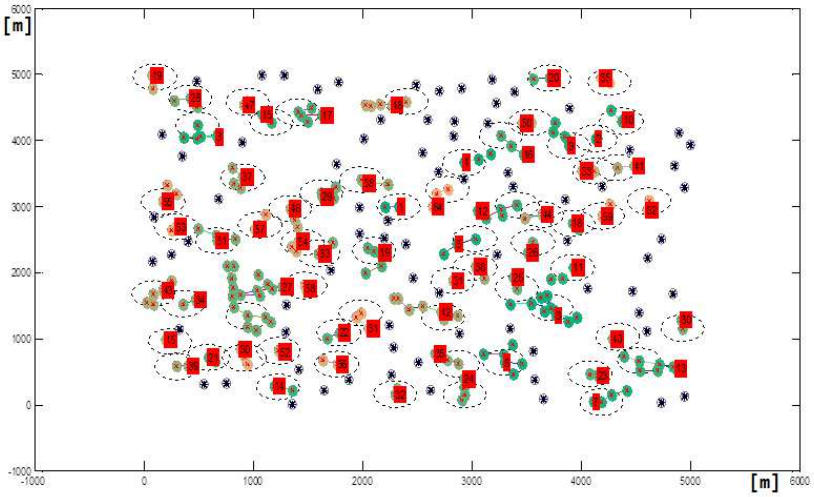

Figure 20: Cluster formation.

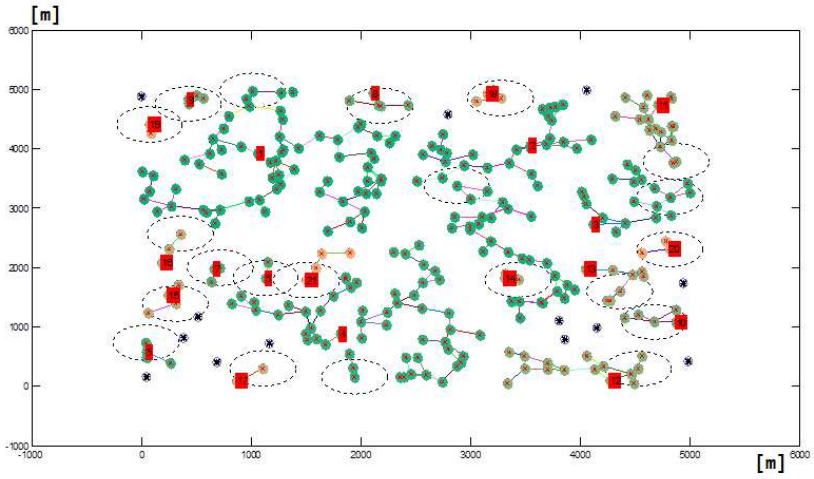

Figure 21: Cluster formation with an improved coverage area.

As we can see, the coverage area of the nodes influences the number of clusters that will form. These new clusters are structures which make the communication among them possible. For a more detailed analysis of these formed clusters, we took the first cluster, and applied a set of different algorithms to it.

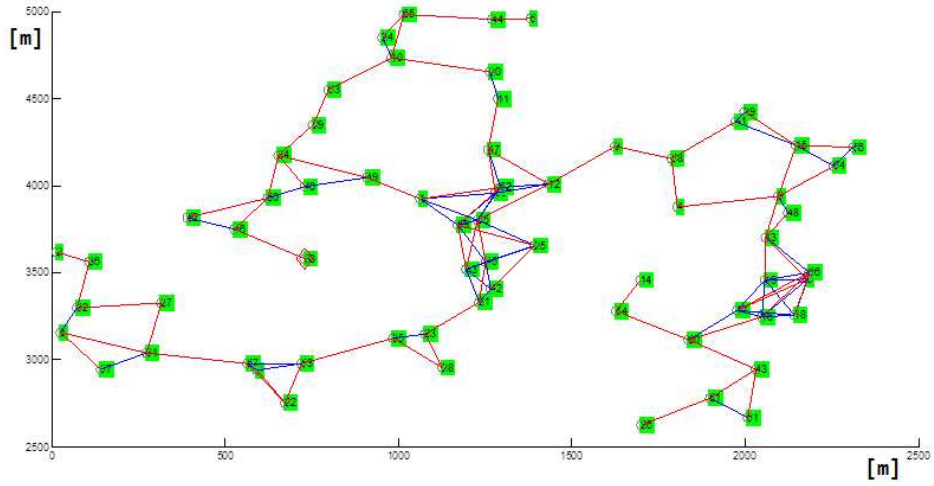

Figure 22: Implementation of Maximum Spanning Tree on a cluster.

We took the cluster with the number 1 index from the previous figure, and made a map with all the possible connections between the nodes with blue lines. Based on these connections, we applied the Maximum spanning tree algorithm, and redrew the edges being part of this spanning tree with red. The source node can be chosen arbitrary. The trees formed by this algorithm are always full trees, because they were interconnected in the first place, and the graph contains all nodes of the system.

Application of the Ford Fulkerson algorithm:

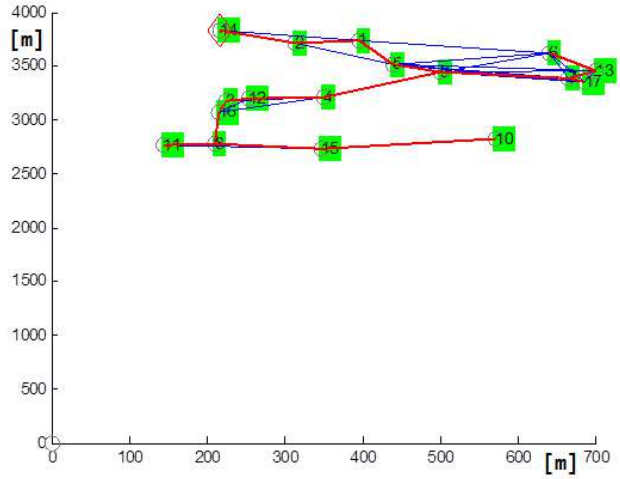

Figure 23: Spanning Tree of the cluster that will be processed.

In the theoretical part we described the algorithm step-by-step. In our simulations it looks the following way:

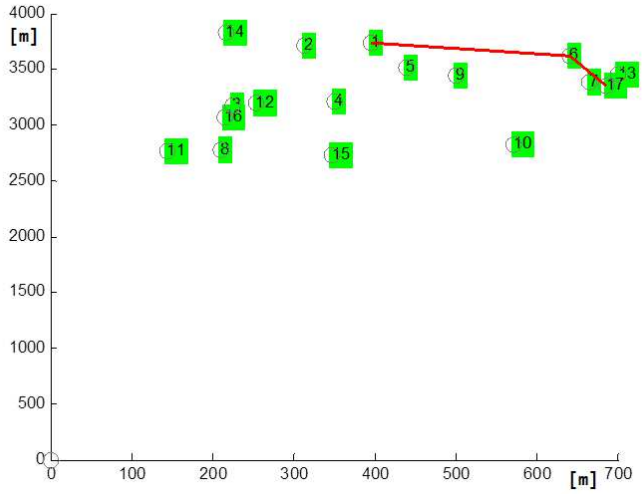

Figure 24: First route found by Dijkstra's algorithm between node 1 and 17.

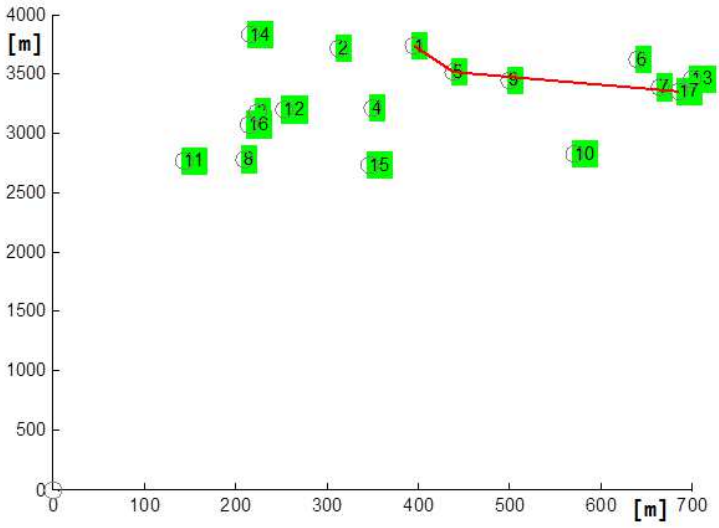

Figure 25: Second route found after an edge was eliminated.

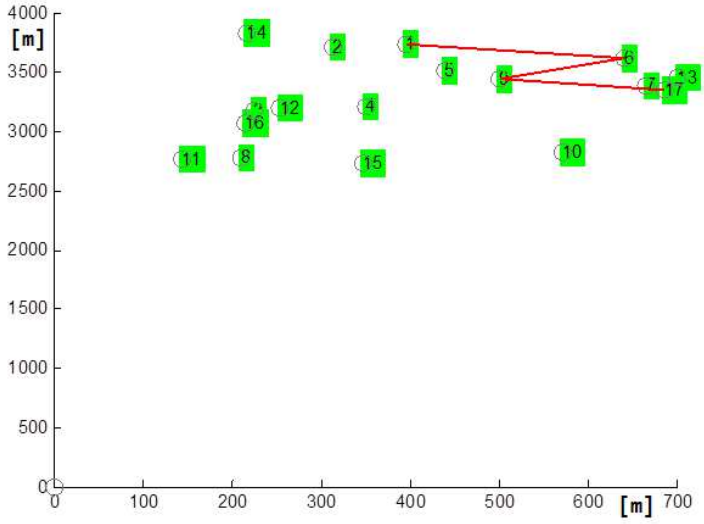

Figure 26: Third path after another edge was eliminated.

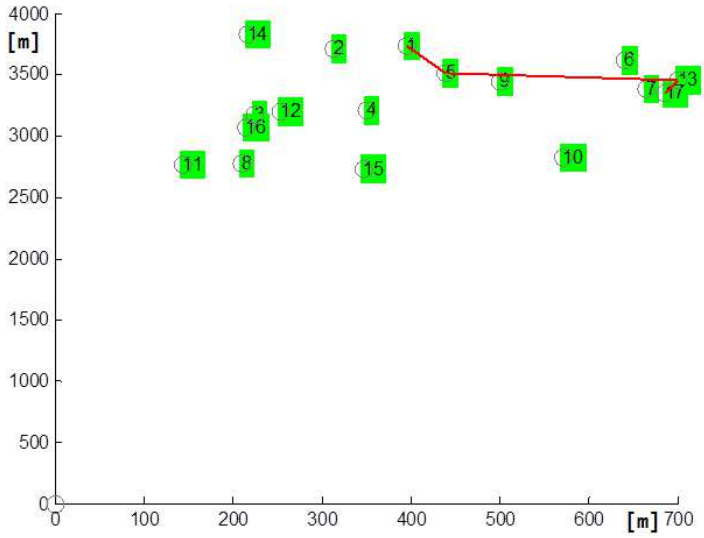

Figure 27: Fourth path after another edge was eliminated.

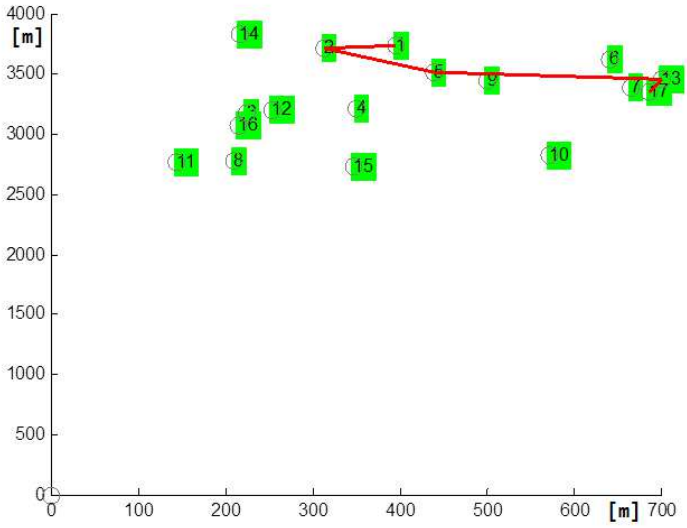

Figure 28: Last possible path, after eliminating the last possible edge.

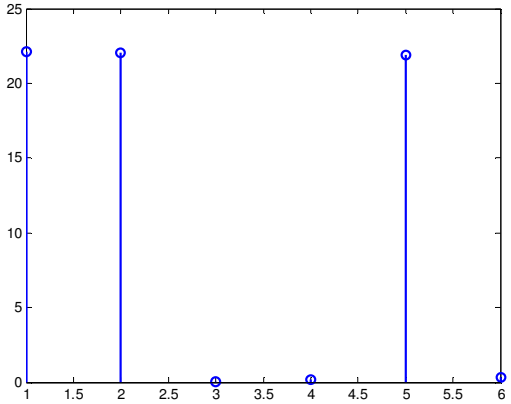

Figure 29: Graph of cuts.

The sum of the cuts gives us the minimum cut, or the maximum flow between the two nodes.

Their sum in the previous example is:

$$22.12 + 22.06 + 0.0093 + 0.21 + 21.92 + 0.3507 = 66.67 \quad (4)$$

The improvement brought to the Ford-Fulkerson algorithm was Dijkstra's algorithm. Instead of depth-first or breadth-first we used the data already obtained with the shortest path algorithm. After eliminating an edge with each step, we updated the tables from the previous Dijkstra algorithm. When we ran out of possible paths, the algorithm ended, and we added up the flows we cut in each step, so we arrived at the maximum flow between the two nodes. To be noted: this transfer rate is a maximum theoretical transfer rate between the two.

We decided to finalize the simulation by adding mobility to the nodes. At this step, we needed to change the position of the nodes. The random speed and random direction was achieved by generating two random numbers, one being the speed and the other the direction. These variables were transformed from polar coordinates to Cartesian coordinates, and added to the initial position. This way we updated all node positions from our system, after which we can reapply the cluster formation algorithms and the cluster processing algorithms.

6. Conclusion

In this paper we proposed the creation of a software platform, that allows the study of the behaviour of nodes in a wireless ad-hoc network, and allows the analysis of these systems by adding new elements to the initial data structures

associated with a node (position, emission power, vicinity list, index of cluster), and with the possibility of choosing the node density of the system, adding new nodes, eliminating nodes or changing the emission power.

After applying the cluster formation algorithm, we applied cluster processing algorithms to the obtained graphs, like: minimum spanning tree, maximum spanning tree, shortest path algorithms, and the minimum cut maximum flow of Ford Fulkerson.

Given the current rate of development in wireless ad-hoc networks, in the future there will be a need for simulation platforms which allow us the study of the behaviour of the network elements, allow us to experiment with routing algorithms, and allow us the analysis of protocols used in these types of systems, networks.

Even if these networks have the capacity to develop in the next few years, they have only acquired attention in the last couple of years. Because of the improvement of technology, the user terminals are getting smaller and smaller and the transfer rates are improving by the day due to system improvements. This branch of communications will meet a great demand in the next couple of years.

As a continuation of this paper, we can consider the possibility of the analysis of routing protocols used in wireless mobile ad-hoc networks (both WSN and MANET), with the help of this platform, from the routing efficiencies point of view. Improvement to the cluster formation algorithms can be made by using more complex algorithms, currently studied by the companies who implement solutions for WSN. Other improvements can be brought by the study of the physical layer, by adding frequency reutilizations schemes to the simulation, and resolving the interference issues in these systems. This aspect was not approached in our simulation platform.

References

[1] Prasad, R., Mihovska, A., "New Horizons in Mobile and Wireless Communications", Artech House Universal Personal Communications series, vol. 4, 2009.

[2] Johnson, D., "Dynamic Source Routing in Ad Hoc Wireless Networks", C*hapter of Mobile Computing, Kluwer Academic Publishers*, 1996, pp. 153-181.

[3] Özcan, A., Zizka, J., Nagamalai, D., "Recent Trends in Wireless and Mobile Networks", Springer Communications in Computer and Information Science series, vol. 162 ISBN: 978-3-642-21936-8, June, 2011.

[4] Anandamoy, S., "Swarm Intelligence based optimization Of MANET cluster formation", *Ms. C. Thesis,* Faculty of Electrical and Computer Engineering, University of Arizona, 2006.

[5] Delavar, A. G., Artin, J., Tajari, M. M., "PRWSN: A Hybrid Routing Algorithm with Special Parameters in Wireless Sensor Network", in *Proc. of Third International Conferences, WiMo 2011 and CoNeCo 2011, Ankara, Turkey*, June 26-28, 2011, pp. 145-158.

[6] Luo, J., Xinxing, G., Tong, Z., Wei, Y., "MI-VANET: A New Mobile Infrastructure Based VANET Architecture for Urban Environment", in *Proc. of IEEE 72nd Vehicular Technology Conference Fall* (*VTC 2010-Fall*), 6-9, Sept. 2010, pp. 1-5.

[7] Lauer, G., "Packet-radio routing", in *Routing in Communications Networks, edited by Martha E. SteenStrup*, chapter 11, pp 55-76, Prentice-Hall, Englewood Cliffs, New Jersey, 1995.

[8] Clausen, T., "A MANET Architecture Model", *Systemes communicants Projects Hipercom*, January 2007, http://hal.inria.fr/inria-00136862.

[9] Hedrick, C., "Routing Information Protocol", Rutgers University RFC 1058, June 1988.

[10] International Standards Organization. "Intermediate system to intermediate system intra-domain routing exchange protocol for use in conjunction with the protocol for providing the connectionless-mode network service (ISO 8473)". ISO DP 10589, February 1990.

[11] Jubin, J., Tornow, J. D. "The DARPA packet radio network protocols", *Proceedings of the IEEE*, vol. 75 Issue 1, pp. 21-32, Jan. 1987.

[12] Perkins, C., "Ad Hoc Networking", Addison Wesley 2001.

[13] Jiang, M., Li, J., Tay, Y. C., "Cluster Based Routing Protocol(CBRP)", National University of Singapore, 14 August 1999, http://tools.ietf.org/html/draft-ietf-manet-cbrp-spec-01.

[14] Perkins, C., "Ad-hoc On-Demand Distance Vector Routing", in *proc. of Second IEEE Workshop on Mobile Computing Systems and Applications WMCSA '99.*, *New Orleans, USA,* 1999. pp. 90-100.

[15] Hong, X., Xu, K., Gerla, M., "Scalable Routing Protocols for Mobile Ad Hoc Networks." *IEEE Networks,* vol. 16. no. 4, pp. 11-21, Jul/Aug 2002.

[16] Wang, Y., Chen, H., Yang, X., Zhang, D., "Cluster based location-aware routing protocol for large scale heterogeneous MANET", in *Proc. of Second International Multi-Symposiums on Computer and Computational Sciences, 2007. IMSCCS 2007.*, *Iowa City, USA,* 13-15 Aug. 2007 pp. 366-373.

[17] Clausen, T., Jacquet, P., "Optimized Link State Routing Protocol (OLSR)." RFC 3626, IETF Network Working Group, October 2003.

[18] Ramanathan, S., "Scheduling Algorithms for Multihop Radio Networks," *in IEEE/ACM Transactions on Networking,* vol. 1, No. 2, 1993, pp. 166-177.

[19] Handy, M. J., Haase, M., Timmermann, D., "Low energy adaptive clustering hierarchy with deterministic cluster-head selection", *4th International Workshop on Mobile and Wireless Communications Network*, 2002. pp. 368-372.

[20] Dijkstra, E., "A discipline of Programming", Prentice Hall, 1976.

[21] Lindsey, S. Raghavendra, C. S., "PEGASIS: Power-Efficient GAthering in Sensor Information Systems", *IEEE Aerospace Conference Proceedings*, vol. 3, pp. 1125-1130, 2002.

[22] Ford, L. R., Fulkerson, D. R., "Flows in Networks", Princeton University Press, 1962.

[23] Chinneck, J., "Practical Optimization: A gentle introduction", textbook Systems and Computer Engineering, Carleton University, 2000, http://www.sce.carleton.ca/faculty/chinneck/po.html.

[24] Cheung, D., Prettie, C., "A Path Loss Comparison Between the 5 GHz UNII Band (802.11a) and the 2.4 GHz ISM Band (802.11b)", January 2002, http://paginadellatecnica.xoom.it/ paginadellatecnica/802_11a-vs-b_report.pdf.