International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 06, June-2016

Gossip Algorithms in WSN with Random or Selective Choice of Neighbor Nodes based on Update Status

Blerina Zanaj
Department of Informatics and Mathematics
Agricultural University of Tirana
Albania

Elma Zanaj
Departaments of Electronics and Telecomunications
Polytecnic University of Tirana
Albania

***Abstract*—Epidemic algorithm applied in Wireless Sensor Network (WSN) is classified as a diffusive type of algorithm due to the way the information is spread in network nodes [1,2,3]. Gossip algorithms as well belong to this class of algorithms and the interest on them is increasing every day more because they offer an efficient way of information transferring through the wide spread systems. It is very important to model their behaviors in order to get an insight view of their performance and of the application built based on them. In this work is tested the performance of two Gossip algorithms through simulations [3,4]. It takes into account the presence of additional packets in the network, the total time spent for the whole system updating and the energy consumed for transmission /receiving of the different kind of packets exchanged between the nodes.**

***Keywords*—Soread Systenm, Knowledge Of Updating Status, Failed Node, Total Updating Time Of System.**

I. Introduction

The increasing interest in WSN is due to their operability as a network compound of a large number of sensors that collect information from the surrounding environment for different parameters. They are used in many applications but at the same time they face a lot of challenges like saving energy, node localization during mobility. For solving this problem different algorithms are developed which help to localize the nodes but they also find the best path with the less possible energy consumption during the communication and information exchange between the nodes. The implementation of those techniques and algorithms in a network there are done with methods called “Plug and Play” [1,2]. The method had made it possible to sensor to be cheap and to have many different applications and functions by changing only their software without intervention into their hardware. During this study will be described Gossip algorithms that are simple to implement. They can fast adapt with size growth of the network. They also have good stability in the presence of different fields of failure like in the process and in links. In Gossip algorithm every node will send data to a random chosen node. In the moment when the destination node gets the packet, it is its turn to choose a random node to send the information. The procedure continues infinitely till the packets arrive in the destination or to the farthest point of network distinguished by the maximal number of hops. In this work are shown two Gossip algorithms that Random (RG) and With Knowledge of Update Status of Neighbor (GKUS) node. These algorithms deliver information through the network in a way to reduce the number of exchanges done in the network. The performance is compared in the terms of total time needed to update the network, extra exchanges and communication and energy consumption [4]. In simulation model is supposed that only one node is the initiator hold the updated information. The initiator is responsible for the starting of updates so that even other nodes take the information. It is supposed that all the nodes have an identification unique number.

In RG a node chooses another node randomly and it decides to update it. All the nodes recognize the neighbor nodes around them but the nodes do not have any clue about their update status. In this situation we expect to have excess messages.

GKUS, the nodes learn about the update status of the neighbor node before they send the information. This is achieved by sending first a request for the nodes status and depending on the response returned is overtaken a concrete action, is sent yes or not an update. When a node is ready to send information, it chooses randomly a destination node like in the case of RG. To the chosen node is first sent a request packet when is asked to respond back with a packet notifying its updating status. If the response from the destination node is positive meaning that yes it is already updated, then the process starts over again with the choosing of another destination node and again a request process restarts. If the answer from the destination node is a no than the source node sends the updating information. In meantime both the nodes addresses are removed from the respective lists of the neighbor nodes that each node contains. This process of random choosing of nodes continues till the list of neighbor nodes becomes empty. As each node is updated only once there are no excess packets but there are information messages in excess. By reducing the most possible the request/confirmation packets and to a fix size is achieved a good performance with this algorithm [3,4].

The paper is organized as follows: Section II presents the system parameters setting important for simulation and results obtained in the work. The algorithm description is shown in section III; in section IV are brought the simulations done testing different parameters; section V brings a summary of the conclusions discussed so far in the work and in the last section number VI are written the references.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 06, June-2016

II. System parameters

The system is the base object where is performed the whole simulation. Inside of it stands the transmitting channel and inside the channel is the nodes. The nodes that participate can vary depending on the simulation machine. Each node has its own identity and it communicates with the other nodes through packet exchange [1,2].

Packets modeling, is make of three fields like: Source(src), Destination (dst) and Information (info), Fig.1.

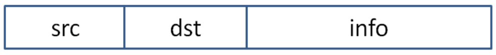

Figure 1. Packet fields

Where:
src- contains the address of source node, size 2B (16 bit);
dst- contains the address of destination node, size 2B (16 bit);
info- contains the updating information, size 4B (32 bit).
Packet size is 8 byte. It is chosen a fixed packet size for being easier to elaborate them from the nodes.

Nodes modeling.Nodes parameters:
Id – identification number of the node
List- the list contains the neighbor nodes addresses
Info- updating information. If it exists it means that the node is updated.

Channel modeling. The channel is modeled as a queue where the packets enter and leave. The channel introduces delays of the packets depending on the errors that may occur. All the nodes have access in the channel. Also the transmitting channel is used for counting the packets that are transmitted and it serves to evaluate the performance.

The wireless communication system between the nodes is composing of two phases: the first phase is the initialization of the values and Gossip algorithms. In the second phase starts the communication between the nodes and their updating.

The changes in Gossip algorithms are shown during the second phase. During the first phase are stabilized the system parameters, the topology is created, the neighbor nodes are calculated to each node and are initiated the nodes and the channel.

The system parameters are set like the number of nodes that will participate during the simulations (numOfNodes), the area of the simulation field (envSize) as well the range of transmitting/receiving of nodes (txRange). Beside these parameters in the network will account nodes failure, during the initialization will be added also the percentage of the dropped nodes (dropped).

Topology creation. The topology is generated by using the rand() function in Matlab. This function generates random numbers in the range 0-1 by using a certain algorithm. Nodes location is done by following the expression below:
n=rand(numbOfNodes)*envSize
Multiplication with the variable envSize makes the value *n* to vary from 0 to the field size and it is one of the coordinates (envSize during the simulation is taken 100).

Neighbor nodes calculation. It is calculated first the matrix of the distances between the nodes as in the formula below:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \qquad (1)$$

After we calculate the matrix of the distances between the nodes, by following the condition that the nodes distances are smaller than the transmitting/receiving range distMatr< txRange , it is calculated the distance matrix, Table 1.

TABLE I. Neighbor matrix

	N1	*N2*	*N3*	*N4*	*N5*
N1	0	1	0	1	1
N2	1	0	1	0	0
N3	0	1	0	0	1
N4	1	0	0	0	1
N5	1	0	1	1	0

The neighbor's matrix has a 1 value when the nodes are reachable and 0 otherwise. Through this matrix are calculated all the neighbors of a node. In the case when a node fails the neighbor matrix will be modified from the algorithm and will create a new matrix that will hold only the effective neighbors of each node, only those who are alive. This way is added only in the GKUS, because the random algorithm does not take the information regarding its neighbor nodes and so it can not find out if a node have died or not. In the simulation done in this work the neighbor nodes are calculated from the base node and not from each node. In the case of the network where some of the nodes have failed the node contains a list with all its neighbors and it compares this list with the other list of failed nodes (calculated randomly by the variable dropped). Each node creates a definitive list of effective nodes that are still active, in network by replacing from the list all the neighbors those that have died.

Channel and nodes initialization. Before it starts the simulation phase a transmitting channel is created where are contained all the nodes and their coordinates. To the nodes by giving them a number *id*, that is unique it is given as well access to the transmitting channel.

III. Algorithms considered

RG(Random Gossip), with random choice has an easy implementation and functioning. A node before it decides to update another node first it checks its neighbors list if there are other neighbors. If there are others than the node checks if it has information to send so if itself if updated. If the node is updated than it chooses another node randomly from its list, creates a packet and sends it. The packet will have as source address the number *id* of the source node, in the destination the address of the other nodes, and information is settled in the info field of the packet. The packet is handled to the channel for its transmission and the id of the destination node will be removed from the list of neighbors in the source node. The get function of this algorithm works like this: firstly the sending node is removed from the list of the destination node, so by deleting it is not needed lately to update that node. Secondly it is checked if the incoming packet has arrived in the node before if so it is a repeated packet. If it is the first time that the destination node receives it than after that it has received it will start the process of sending it to the other neighbors of it. The process goes on till all the node list of neighbors will be become empty.

GKUS, node control function differently from the RG, because it firstly it is notified about the updating status of the chosen destination node. Depending on the response from the destination it is decided what will do. The informing process or knowledge of neighbor node status is done from the node control function. Firstly is created the request packet, that is sent to the channel for transmission and is awaited for the ending of sending/receiving process. After this process the neighbors list of the node is checked. If in the list stands again the destination address of the node then no updating is sent toward it, elsewhere the destination node can be chosen to be updated.

Send/Receive function. In the algorithm with GKUS after having controlled the neighbors list and the updating information, the node chooses randomly and checks through the control node function if this node is already updated. If the node is not then the process is the same one as the send function of the RG. If the chosen node is not updated then the source node creates an updating packet, where the info field is set equal to the info field of the source node. The packet is sending for transmission into the channel. At the end the list is updated and the process restarts all over again. Receive function checks the information that is kept in the info field of the packets. The info field of the packets is considered as updating information when it is set in the range of values 0-100, request when it 200, confirmation if the destination node is updated (info=300) and when it is not updated (info=400). The values are chosen not to similar in values in order to dismiss any probability of getting wrong information if one or more bits are inversed during the transmission. The get function depending of the value received in the info field decides what to do. If the received packet is a request the node checks its update. If it is updated it responses with a confirmation packet where the info field is 300. Otherwise the node does not send a packet or its response set the info field to 400. It is chosen like this because we want that the no answer and the absence of a response the action performed in the source is the same the destination node is not removed from the neighbor list. If the information of the received packet is a confirmation (yes/no) the node continues with the updating of the neighbors list if the info is set to yes, in the case when it is no it does not take any action. If the packet contains updating information then as in the case of random algorithm, the node updates the list of its neighbors, looks for packet repetitions then starts with the sending process. The only problem in the simulation with such algorithms is that nodes send their packets one after the other. This is due to the programming where the processes are executed in a row and not in parallel as it really does in reality. In the physical environment it may happen that many nodes transmit in the same time and the confirmation (yes or no) may happen before the updating of that node. So in the physical environment there is always an uncertainty in defining nodes updating. From the other hand the functions in programming follow a row and like this they know for sure if there are executed or not a certain structure.

IV. Simulation

Simulation parameters. The simulation is done for 50, 100, 150, 200 nodes with a transmitting radius 10, 15, 18 and 20 units. The dimension of the square field, where is done the simulation it get the dimension of 50, 100,150 and 200 units. Transmitting channel includes some delays in the communication between the nodes. The delay time assumed is 0.001s (1ms), which may change regarding the real transmitting values. The node dropping may change to 0%, 10%, 20%, or 30% in both algorithms.

Random Gossip algorithm. Random choice Gossip algorithm (nodes failure from 0% to 30%) is shown as in the Fig 2 and Fig.3, while nodes and network parameters are like in the Table 2.

Table 2. System parameters

txRange	10
numOfNodes	200
envSize	100
dropped	0% dhe 30%

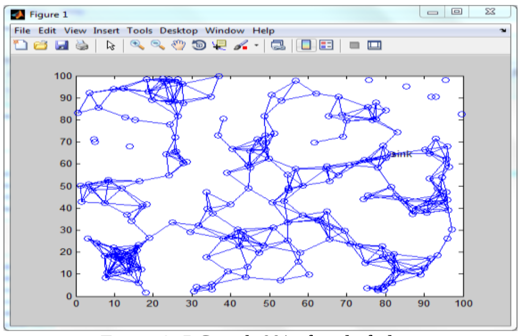

Figure 2. RG with 0% of node failure

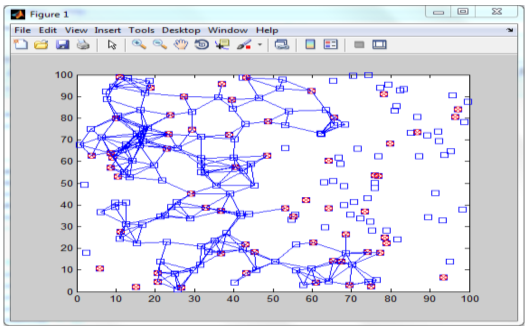

Figure 3. RG with 30% of node failure

As Fig.3 shows the nodes get updated more than once and this create packets load in the system. At Fig.3, nodes in red are those who have failed due to their energy consumption. In this way those do not contain information and can not send information toward their neighbor nodes. In this case it is obvious that the nodes in this network do not know the failed node and as consequence those try to send information toward them that will result in information loss and increase the load in the network.

The GKUS, with node failure of 0% and 30% is shown in Figs. 4 and 5. Differently from the RG with this algorithm the nodes are not updated more than once and even in the case of the network where the nodes may fail, each node knows with whom of the neighbor nodes it can communicate with (no knowledge which of the nodes have remained alive).

Bye doing like this the energy is conserved because there will be no control packets sent toward the failed nodes. Nodes failure notification in this algorithm is performed through the algorithm check nodes. But even this algorithm has packet loads, due to information exchanged like request/confirmation between the nodes. In Fig.5 are shown only the updating packets. The exchange packets for request/confirmation are not shown because the picture will be more complicated and less comprehensive.

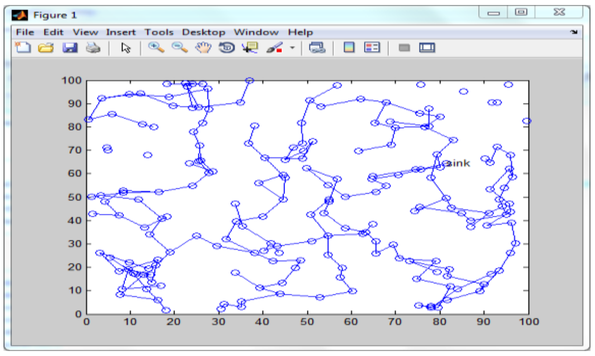

Figure 4. GKUS with 0% of node failure

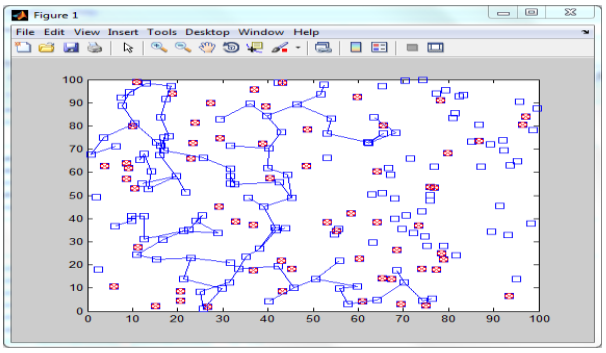

Figure 5. GKUS with 30% of node failure

Analysis of simulation results. The comparison between the two algorithms is done for four different percentages of dropped nodes (0%, 10%, 20% and 30%) to see how the performances with the increasing of the failed nodes. Another comparison is done for the same algorithm for the four different percentages of failed nodes. In such way we can put in evidence clearly the difference of the two algorithms and the impact of node failure as cause. The measurements are done for a 2D space with size envSize=100x100, the number of network nodes changes with the parameter numOfNodes= 50, 100, 150, 200. The transmission range is unchangeable txRange=10 and the percentage of the nodes that fail in network change in percentage with the parameter dropped=0%, 10%, 20% and 30%. Results shown below are as result of averaging of 50 measurements in 50 different networks that were generated randomly. In order to compare the two algorithms the measurements are based in transmission time, the energy consumed during the transmission and receiving and the network load.

Total time of updating. Total updating time is the time measured from the start till the end of the simulation phase. This time is measured in seconds and it depends on the delay introduced from the transmitting channel, Fig.6. As it can be expected the total time of updating of the system with the GKUS is higher and this because to the high excess informative packets not updating packets. Both algorithms have quasi a linear growth with the growth of the packet numbers. The comparison of the algorithms with each other, is done also between the algorithm with itself during the different percent of nodes failure.

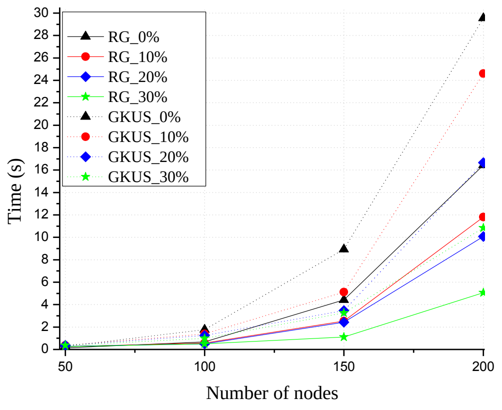

Figure 6. Updating time of network with different % of failed nodes

Total updating time of the system depends linearly from the delay introduced from the system. In simulation of both algorithms there is a delay of 0.001s introduced from the transmitting channel.

By observing the graphics there are two phenomena:

It is observed a general decrease in transmitting time with the growth of percentage of failed nodes for both algorithms. This is so because with the failing of the nodes decreases the number of effective nodes in network, as a consequence also the message exchange in network nodes will become faster. With the failing of 30% of the nodes in the system the updating time will become 3 times shorter compared to that of the whole system. It is observed a decrease in the differences between the graphics lines of the GKUS and that of RG as observed in Fig.6. Actually the difference in time between the two algorithms in the case of 200 nodes varies from 14s (where in the network are all the nodes) to 6s (where 30% of the nodes have failed). This is due to the informative packets that are exchanged in GKUS. Data sending in all the nodes (also to those who have failed) in the random gossip algorithm cause a network overload. The overload is excluded in the GKUS where is excluded the possibility to send heavy packets in the network (as GKUS firstly are sent only the informative packets), consequently the communication goes faster. The data obtained from simulation shows once again that the gossip algorithm with knowledge when the nodes percentage that is alive decreases adapts better.

Network load. Redundancy is made of received packets in random gossip and as request/response packets in the gossip with knowledge. The result obtained is shown in Fig.7. It is observed that the GKUS have higher load of packets. This due to the continuous communication between the nodes with request packets, from which then is awaited response packets. In the algorithm with random choice there is only one redundant packet, the one that can be sent to the destination node even that the destination node may be already updated. Instead in the case of the GKUS there are the request and the response from the destination.

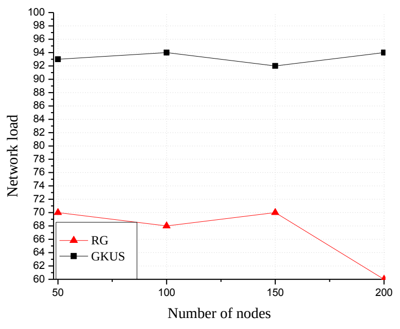

Figure 7. Network load for both algorithms

So there is one more packet in the least case. But in both algorithms there is a stable redundancy of packets. In the case of the RG choice of the node there are 60% of all nodes instead in the case of gossip with knowledge there are near to 95%. This shows that both algorithms conserve a certain level of unwanted or information packets and this is independently of the number of nodes. There is a slight decrease in the RG that shows that this algorithm adapts better in big networks.

Consumed energy. The energy consumed is the total consumed energy from the sending and receiving of the packets. Firstly the nodes are started with a set value of transmitting energy (txPower) and receiving (rxPower) of 100 units. It is supposed that each sent and received packets consume 3units of the starting energy. The values assumed only for the simulations and are seen in percentages. If there are required more exact values than the parameters will be adapted to the actual standards of wireless sensor networks.

Energy consumption from the transmission. The results obtained from simulation are shown in Fig.8, at the beginning there is done the comparison of two algorithms between the two for node failure of 0%, 10%, 20%, 30% and then the algorithm are observed for nodes failure by themselves. In the case when there is no node failure we can observe that the GKUS asks for more energy for packets transmission in the network and this energy increases with the increase of the number of nodes.

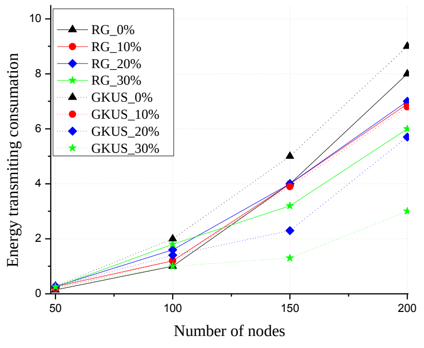

Figure 8. Consumed energy from packets transmission for different % of failed nodes

This is due to the updating messages sent from this algorithm and those not only overload the network but as well are expensive from the energy consumption point of view.

In this case it is observed that the algorithm behaves more or less in the same manner from the energy consumed for transmission [3]. The maximal level of energy consumption goes to 8% in the case of the network made of 200 nodes. This is so because the RG at this point starts to show its disadvantages in information packets sending even toward the failed nodes, but this is excluded in the algorithm with knowledge of the updating status. This fact causes that the difference in consumption in both graphics to decrease. At this point there is done the comparison of the algorithms with each other.

So it results that the energy consumption in transmission grows with the growth of the number of nodes in the network. This comes from the fact that more nodes bring more packets transmitted in the network. Although this consumption continuous to be relatively low. The GKUS behaves better than the RG when there are some of the nodes that have failed. This is observed in Fig.8, where the energy consumption in transmission in the GKUS stands below the curve of the RG. In the case of nodes failure to 30%, the energy consumption for transmission for the random gossip reaches the double of the energy consumed in the case of the gossip with knowledge of updating status (the first in 6% and the second in 3%). This is because of the sending of informative packets in the algorithm with knowledge that avoids sending the heavier packets needed for information exchange toward the failed nodes by decreasing like this the network load.

Energy consumption for transmission decreases with the increase in number of the failed nodes [3]. In the GKUS the nodes failure of 30% cause that energy consumption become 3 times smaller compared to the network where all the nodes are still active in network (from 0% to 3%). With the increasing percentage of failed nodes there will be less information passing in the two algorithms, as consequences less packets exchange in network.

Energy consumed from the receiving. Results obtained for energy consumption during the receiving, Fig.9. GKUS spends more energy in receiving than the RG. This is so because of the response packets received from neighbor nodes. Fig.9 shows the energy consumed in both algorithms in the case where there are some different % of nodes failures.

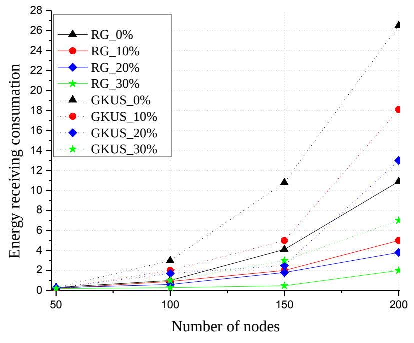

Figure 9. Energy consumption in receiving for different % of failed nodes

In the case of the network without node failure, it can be observed that the GKUS needs more energy to receive the packets in network and this energy increases with the increase in the number of nodes. This is due to updating messages of this algorithm that are received from the nodes and they overload the network but those are heavy from the energy consumption point of view[3-4].
So, the nodes failure will vary in percentages and conclusions will be brought regarding each algorithms deployment and how will vary the energy consumption from packets receiving in case of increased number of nodes or in the case when their number decreases due to nodes failure percentage variation.
So it results that the energy consumed in receiving increases with the increase in the number of nodes in network. This is due to the fact that a bigger number of nodes bring a bigger number of packets transmitted in the network.

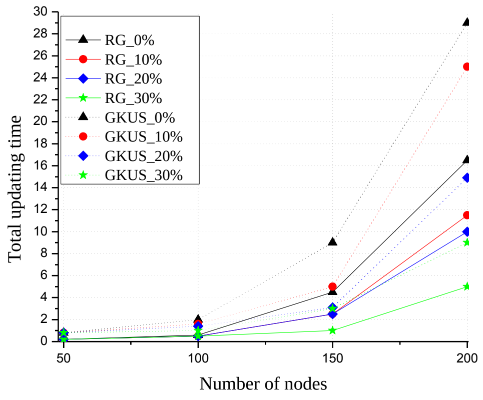

Figure 10. Total updating time for different % of failed nodes

Random algorithm have a better behavior toward energy consumption in receiving compared to the algorithm with knowledge of updating status when there is an increased number of nodes in network. So when the number of nodes in network is big it is more convenient to use RG algorithm. The GKUS behaves better than the RG in the case when some of the nodes have failed and this can be seen in Fig.10, where the graphic of energy consumption in receiving the curve for the gossip with knowledge stand below to that of random gossip. This is so because of informative received packets from the gossip with knowledge avoids the packet receiving those that are heavy that are for information exchange without forwarding them toward the failed nodes and by doing like this the network load is decreased.
Energy consumed during the receiving decreases with the increasing in number of the failed nodes in network. It results that in the algorithm with knowledge of the update status (figure 26), with 30% of failed nodes the energy consumption decreases with 3.25 times compared with the case where in the network all the nodes are all active (from 26% to 8%). This is so because with the increase of the failed node in number, there will be less information transferring in both algorithms, so less packets exchange in network.

Network Connectivity. There is a logical connection between the network connectivity and transmitting range in the system. The simulation are performed on more than 20 different topologies of the network, those are generated randomly and the average of those 20 measurements to have a summary graphic of those averages. The environment where are performed the measurements is in 2D with an area of 100x100 and transmitting range (txRange) that varies from 1,4x100, decreases with a 0.2 step. For each transmission range of values there is an average number of neighbors node in network. The measurements are done with 50,100,150 and 200 nodes and for each of the cases the nodes failure percentage varies from 0%, 10%, 20% and 30%. The results are shown in Fig.11.

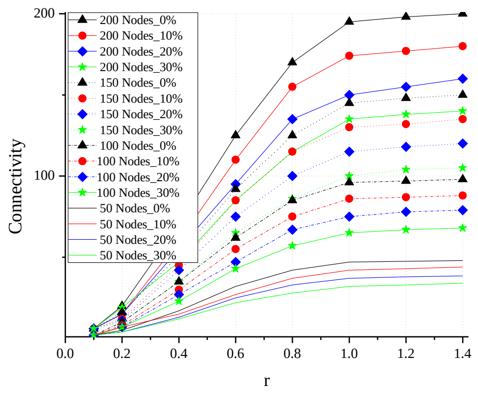

Figure 11. Connectivity for different % of failed nodes

As Fig.11 shows, that the decrease in the number of neighbors for each node is achieved when: there is an increase in the percentage of failed nodes. So for example it is distinguished that in the network without failed nodes the maximal number of neighbor nodes varies in 49 for each node, instead in the network with 30% of failed nodes this number reaches 34.
The transmission radius is shortening in all the graphics. The main impact the transmission radius is in the case when it is decreased in values from 0.6 to 0.4. For example: the radius is reduced with 0.2 unit so the number of neighbor nodes is 38 less than the case with 30% of failed nodes, 43 in the case with 20% of failed nodes, 48 in that with 10% of failed nodes, and 54 in the case when there is no failed nodes in network.
The number of nodes is reduced as seen, Fig.11. So it results that network become fully connected in the case when the transmission radius become 1.4 of one of the dimension of area of transmission. From the last figure you can observe that for this transmission radius a node in the network of 200 nodes have approximately 199 neighbors (so it can communicate with all except itself), the one with 150 nodes have 149 approximately and so on.

V. Conclusions

By simulation with matlab of both algorithms of rg and gkur it was checked their performances regarding different issues like excess packets in the system, total updating time, energy consumption for transmitting/receiving. all these cases were checked depending on the number of nodes in network and the different percentages of failed nodes in the system. Beside these parameters the algorithms were analyzed for the transmission radius and how is its impact together with the different percentages of nodes failure in the network connectivity. from the simulation there is an improving tendency in the gkus if we compare it with the rg with the increase of the number of failed nodes in system. The difference in time between the two algorithms in the case of 200 nodes varis from 14 s (where in the network all the nodes are active) to 6s (where in the node 30% of the nodes have failed). in the case of consumed energy in receiving and transmitting, the consumption is 3 times lower while the failed node percentage varies from 0 % to 30%. it could be observed that the increase in the number of nodes that have failed combined with the decrease in transmission radius will bring a decrease in the number of neighbor nodes for each node. but from simulation there was also an excepted result when the number of nodes is 50 when all the nodes are active the maximal number of neighbors is 49, instead for the same transmission radius while the number of failed nodes reaches 30% the number of neighbor nodes decreases to 34.

VI. References

[1] R. Bakhshi, Gossiping Models ,“ Formal Analysis of Epidemic Protocols’’.

[2] M.A. Matin and M.M. Islam , “Overview of Wireless Sensor Network” , Wireless Sensor Networks – Technology and Protocols, (Ed.) ISBN 978-953-51-0735-4, INTECH Publisher, Croatia, 2012

[3] Giuseppe Anastasi,Marco Conti, Mario Di Francesco, Andrea Passarella “Energy Conservation in Wireless Sensor Networks: a Survey”, Ad Hoc Networks 7 (2009) 537–568p.

[4] M. Baldi, F. Chiaraluce, E. Zanaj, “Performance of gossip algorithms in wireless sensor networks”, Solutions on Embedded Systems, Spinger Science + Business Media, First Chapter.pp. 3-16, 2011. ISBN: 978-94-007-0637-8.[5] M. Simek, P. Moravek, J. Silva, “Wireless Sensor Networking in Matlab Step-by-Step” , January 2010.

[5] MathWorks Documentation - MATLAB V7 Introductory and Programming

[6] R. Prasad, A. Mihovska, “New Horizons in Mobile and wireless communications”, Artech House, 2009. 447 p.Publication: Anthology

[7] Simulcast Emulation for Matlab , https://code.google.com/p/simulcast/