Annales UMCS Informatica AI XIII, 2 (2013) 49–67
DOI: 10.2478/v10065-012-0041-z

Annales UMCS
Informatica
Lublin-Polonia
Sectio AI
http://www.annales.umcs.lublin.pl/

Simulation of reconfiguration problems in sensor networks using OMNeT++ software

Paweł Dymora[1*], Mirosław Mazurek[1†], Piotr Płonka[1‡]

[1] *Rzeszów University of Technology, Faculty of Electrical and Computer Engineering, Department of Distributed Systems, ul. Wincentego Pola 2, 35-959 Rzeszów, Poland*

Abstract – The paper presents a comparative analysis of reallocation algorithms in a structure of wireless sensor network in the event of a failure of network nodes. The article contains detailed research results of wireless sensor networks technology, with particular emphasis on thte network-layer protocols - routing protocols. In the research the simulation environment OMNeT++ was used to study the properties of reconfiguration and reallocation problems in the wireless sensor networks.

1 Introduction

Increasingly, wireless sensor network technology appears in many areas of life. Automation particularly in the areas of critical infrastructures, responsible for warning of the occurrence of fires, tsunamis, landslides or floods is increasingly entering into our daily lives. The systems monitoring the water level in the rivers, with the wireless sensor network technology allows precisely and accurately to check the water level in a very large area and in the event of a warning value overrun - without human intervention can trigger alarms in the affected area. Where the collection of accurate information is problematic due to the hazardous environment, or the need to remove people from the area of research - the application of wireless sensor networks is the ideal solution. In addition, the wireless sensor network can be applied in military solutions, monitoring animals and environment pollution, to control the device parameters, monitoring the health of patients or building automation [**1**].

*pawel.dymora@prz.edu.pl
†mirekmaz@prz.edu.pl
‡pplonka@prz.edu.pl

Possibilities of wireless sensor networks are enormous and their use is limited only by the imagination. This paper describes the most important feature of wireless sensor networks which is the ability to act independently without human intervention. For this to happen the network must respond to changes in topology, which can be caused by network nodes (sensors) failure, lack of energy in the node or electromagnetic interference. Since such a network in most cases is left itself, the human intervention in network elements after its allocation in the terrain is minimal, the network must be fault tolerant. In such a structure, there is a usually rigidly defined path providing measurement data to the users, everything happens dynamically in real time. Therefore, an important role in the wireless sensor network technology is played by self-reconfiguration protocols by which network responds to changes and faults, adapts and effectively manages the available resources. Self-reconfiguration protocols, which can also be called routing protocols, are a common topic of research and scientific work of scientists around the world. An important aspect of the design and development of routing protocols for the wireless sensor networks are relevant and meaningful simulations using network simulators [**2, 3,** 4].

2 Wireless sensor networks

Wireless sensor network is an infrastructure composed of measuring nodes, data processing and communication elements that allow users to monitor and react to events and phenomena in a given environment. Sensor network is composed of four basic components: sensors allocated in a distributed system, internal network connecting the sensors (usually wireless), the central point of grouping information and a set of modules responsible for the processing and exploitation of data. As a sensor network node we understand a sensor with an integrated controller responsible for the data processing and sending collected information to a central network point via the internal network - usually wireless [**3, 4, 5**].

2.1 Sensor network topology

Wireless sensor network architecture is a distributed architecture. This means that the sensor network is composed of processing units – sensors, whose each has a separate processor, local memory and I/O module. Because sensors do not have shared memory, they communicate with each other, creating a distributed communication network. The nodes of the given site of action, send (directly or indirectly) collected and partially modified information to the parent node. The parent node via the Internet, communications satellite or cable transmits the data collected from the sensors field to a central unit which is responsible for processing, interpretation, and presentation of information to users. Fig. 1 shows the approximate sensor network architecture diagram [**2, 1, 6**].

Wireless sensor network may consist of a large number of sensors, densely deployed in a given area. When using a fixed logical topology, programmable before deployment

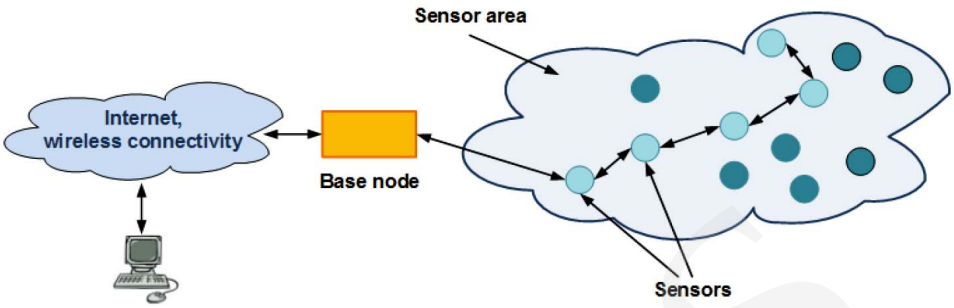

Fig. 1. Sensor network architecture model.

in the field, we should manually deploy the individual nodes to maintain adequate distance between them, such cases are very rare and used with a small number of sensors. In most cases, during the deployment of network nodes the opportunity to drop sensors from a plane over the desired area of deployment is used. Also prone to failure of individual sensors makes it difficult to manually configure logical topology. For this reason, the logical topology of sensor network is constantly changed, and thus its management is a demanding task. The routing protocols deal with that automatically without human intervention, they configure the logical network topology and adapt to the changing number of sensors or a failure in communication between them. Depending on features, the individual nodes of a sensor network may vary. The sensor node type selection largely determines the sensor network action, but the main modules, which each of the sensors consists of are the same. The general diagram of sensor architecture is shown in Fig. 2. However, in Fig. 3 there is shown a diagram of sensor network node software architecture [**2, 1, 6**].

Wireless sensor consists of a *measuring module* - this module is the most important module in the node and is involved in obtaining data from the environment for further processing. *Computation module* - is responsible for pre-processing the data acquired by the sensor. It consists of a microcontroller with a memory, equipped with interfaces and permanent record storage, sends and collects the necessary information of the entire node. *Communication module* - is responsible for communication with the other sensor nodes within the antenna range. It sends and receives information wirelessly using radio waves, infrared radiation or by the use of the optical medium. *Power supply module* is the most critical module in a node. It usually consists of a battery, which causes that a single sensor lifetime is strictly limited to the capacity of energy [**2, 5, 6**].

In most currently used sensors, it is impossible to replace or recharge the energy to prolong the life of the node. Therefore, to make the rational use of limited energy resources, special schemes and providing energy-efficient communication protocols are implemented. Also research on the use of solar energy and vibration in order to gain extra energy are conducted. Generally, the sensor has the following software subsystems: *operating system* is responsible for the management and monitoring of all components

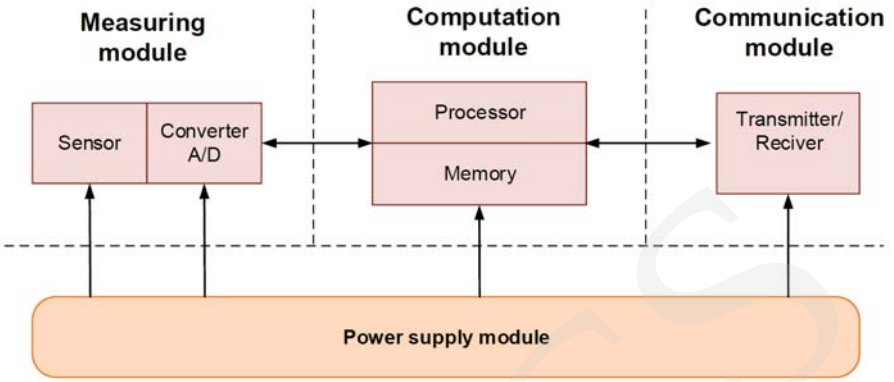

Fig. 2. Wireless sensor node architecture.

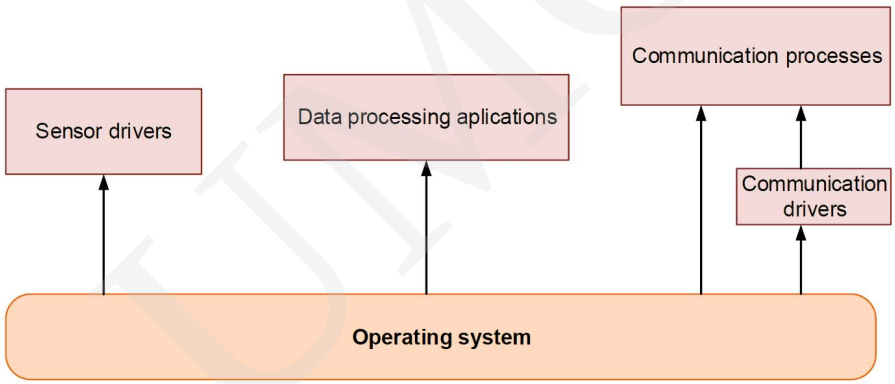

Fig. 3. Sensor software architecture model.

of a sensor network node. At present time, the most commonly used operating system in the sensor platforms is TinyOS. *Sensor Drivers* are software modules that manage the basic functions of the sensor. *Communication processes* support communication in the transport layer, network and data link layer. This subsystem is responsible for the management of node communications functions, including routing, buffering and forwarding packets, responding to changes in the topology and encryption. *Communication drivers* are software modules that support coding and physical layer. They conduct management of the transmission in the transmission channel, including time synchronization, signal encoding /decoding, signal modulation, mystified bit recovery method and bits counting. *Data processing mini-apps* are applications for numerical processing and storing of data prepared for transfer. Such applications include all kinds of simple programs that allow to obtain core values from the gathered data, but they are sufficient to use single node hardware resources [**2, 5, 6**].

2.2 Factors influencing the sensor network topology designing process

Wireless sensor networks as a type of special-purpose computer network require consideration of many factors during the design process. The most important design criterion for sensor networks is to determine the high level of reliability (fault tolerance). Due to the nature of the operations, wireless sensor networks are used in difficult and unpredictable environments, where some of the nodes may be damaged, or may be environment interferences or simply a lack of energy which will result in blocking the node action. Destruction of single nodes or groups of nodes in wireless sensor networks can not affect carrying out the tasks, which involves using a factor that determines the ability to meet desired functions in the network. Such a factor is the fault tolerance, the amount of potential losses in the number of nodes that do not damage the correct and reliable operation of the entire network. Wireless sensor reliability is determined based on the Poisson distribution (1). It is defined as the probability of failure of the sensor node over the time where λ_k is a coefficient of k nodes failure at the time t **[2, 7, 4, 5, 6]**.

$$R_k(t) = \exp(-\lambda_k \cdot t) \tag{1}$$

Another criterion is the number of sensors. Wireless sensor networks can be deployed on a large area, the number of network nodes can range from tens to thousands, depending on the network tasks. Therefore, designing applications and protocols for the sensor network we need to take into account the large number of nodes. Distribution of nodes in a given area is usually random (e.g. drop from an aircraft) and in order to calculate the density of nodes we use the formula (2) where N represents the number of network nodes in the area A, and R denotes a node transmission range. It allows to present the number of nodes within the transmission range **[8, 7, 9, 5, 6]**.

$$\mu(R) = \frac{N \cdot \pi \cdot R^2}{A} \tag{2}$$

Communication in wireless sensor networks is as the name suggests through a wireless media. The wireless medium are: radio waves or optical waves. The effective and resistant to interference data transmission between the sensors they should apply the coding and modulation schemes. Therefore, an appropriate choice of transmission medium is another criterion influencing the designing of sensor networks. Due to the fact that the wireless sensor networks consist of a very large number of sensors, the cost of producing a single node is of great importance in the design of a sensor network, therefore, an important criterion is the cost of production. Another, one of the most important factors taken into account in the design of wireless sensor networks and the protocols as also algorithms are used in them, is the criterion of power consumption. Since the sensors are wireless, they do not have a cable connection to an external unit providing energy, thus they are equipped with its own limited power supply. By preventing the transmission of data between sensors, because of a lack of energy, operating

the entire network can be threatened, or at least the reorganization of data transmission path in the network will be required. Therefore, saving energy consumption of wireless sensor network nodes is a task of great importance in the designing process [**8, 7, 9, 5, 6**].

2.3 Self-reconfiguration algorithms

In order to make use of all potential advantages from the use of wireless sensor networks a very high self-organization and coordination between all the sensors is required. Routing protocols play a major role here in creating a wireless multihop network that can be self-organized, self-reconfigured and fault tolerant. Sensor networks due to the specific application require specific routing algorithms that are very different from those used in computer wireless networks. Routing protocols for wireless sensor networks can be classified according to three main common features: formation of the transmission path, type of network structure and communication initiator. Categorization according to the formation of the transmission path is divided into protocols: proactive routing protocols, reactive routing protocols and hybrid routing protocols. Classification by the network structure: flat network structure, also known as a unitary structure (Fig. 1), hierarchical network (Fig. 4) and direct network (Fig. 5) [**7, 1, 5, 6**].

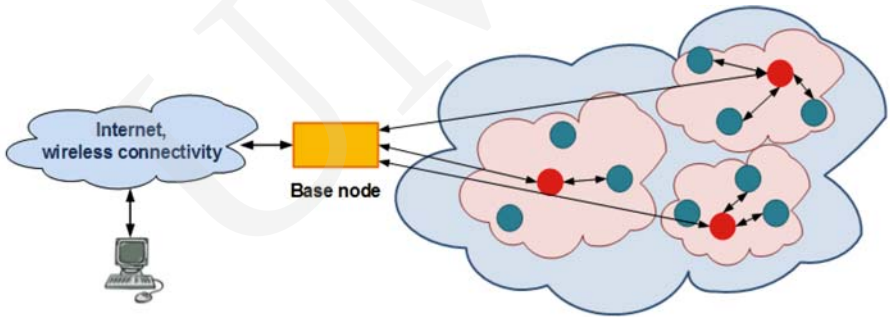

Fig. 4. Hierarchical network.

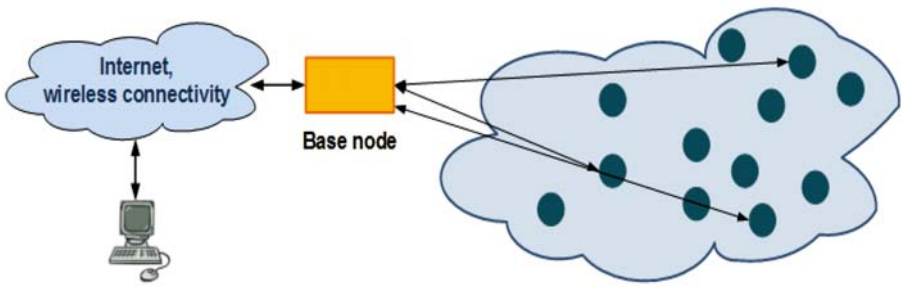

Fig. 5. Direct network.

Another categorization according to the communication initiator is protocols initiated by the source and the protocols initiated by the destination [**1, 5, 6**].

One of the most important protocols in flat self-reconfigured sensor networks is the AODV protocol *(Ad-hoc On-demand Distance Vector*). This is a reactive protocol, it is very rapidly discovers routes and does not force the network nodes to keep information about inactive sensors. It responds quickly to changes in the network topology. Paths in this protocol are marked with special sequence numbers, so that by using the right kind of application it avoids route loops. If the sensor needs to send data to the parent node or another node on the network and does not have a route in the routing table, the attempt to detect the route is taken. Another important protocol is DSR (*Dynamic Source Routing*). It is a reactive protocol very similar to algorithm searching for a route implemented in AODV. The main difference compared to the AODV protocol is a way of storing data about the route. In the case of AODV in the routing tables the information about the following hops is stored in this protocol, the full details of the route from the source to the destination are stored. Other protocols that can be mentioned are: the SPIN protocol (*Sensor Protocol for Information via Negotiation*), Directed Diffusion, Energy Aware Routing protocol [**1, 5, 6**].

In the hierarchical sensor networks, we can distinguish the following self-reconfiguration protocols: LEACH (*Low-Energy Adaptive Clustering Hierarchy*), OLSR (*Optimized Link State Routing Protocol*), Teen (*Threshold-sensitive Energy Efficient Sensor Network Protocol*) and APTEEN (*Adaptive Periodic Threshold-sensitive Energy Efficient Sensor Network Protocol*) and HAR protocol (*Hierarchy-based Anycast Routing*) [**1, 5, 6**].

3 Wireless sensor networks testing environment – OMNeT++

There are many programs to simulate computer networks, e.g.: NS-2, SENSE, TOSSIM, JSim or OPNET. However, one of the best in the case of application with the wireless networks especially sensor networks is OMNeT++ (*Objective Modular Network Test-bed in C++*) which is an open source network simulator written in C++. It is implemented in an object-oriented architecture, divided into various modules. The project OMNeT++ was started in 1993 and was not originally designed as a computer networks simulator, in fact it was generally used as a discrete event simulator. Thus, it contains a large number of simulation models [**3, 4, 10**].

OMNeT++ simulator consists of simple and complex modules. Complex modules consist of simple modules which can be nested. Adding to OMNeT++ software simulation models (frameworks) such as INET, it becomes a powerful wireless sensor network simulator. Research on performance of similar software showed that OMNeT++ is one of the best systems in the context of memory usage and short simulation time. We can fully modify the whole environment in which the simulation is running, and even include the simulation of OMNeT++ into larger applications. Library of component

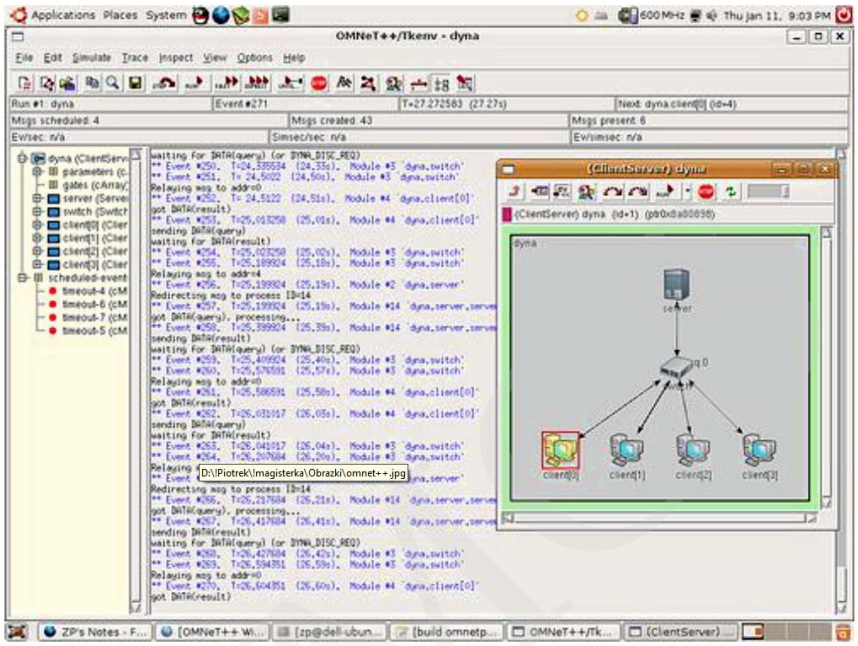

Fig. 6. Simulation screen in OMNeT++.

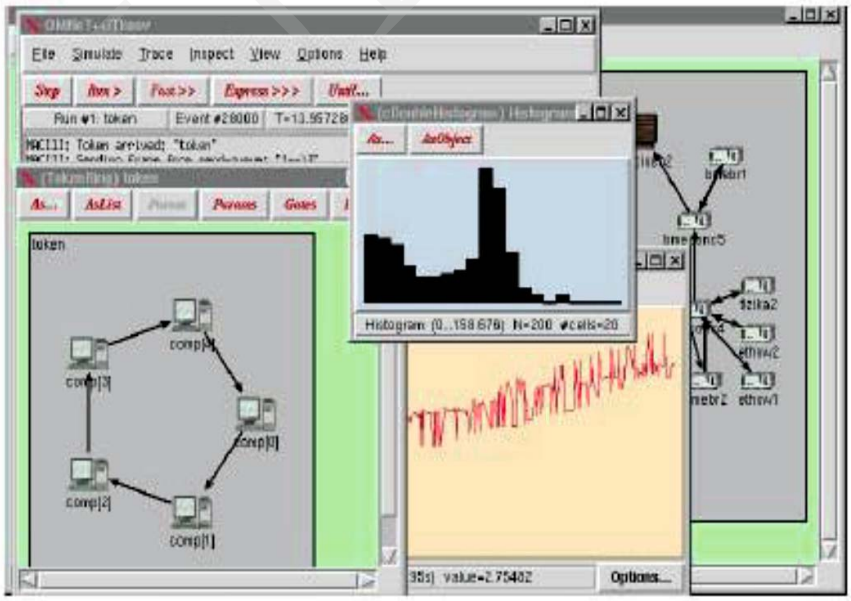

Fig. 7. Screen of the user GUI (Tkenv).

models (called *Model Component Library*) is composed of complex and simple modules compiled code. Gateways are the module I/O interfaces. Connections between gates can be assigned to parameters responsible for e.g.: delay, data rate, error rate. Structure which consists of various modules is defined by the language NED (*Network Description*), which has a simple but well-developed syntax for creating different nodes and topologies. The environment provides the ability to create structures by using a text or graphical user interface. The user interface is defined by the library ENVIR. OMNeT++ provides two user interfaces: Tkenv and Cmdenv. Tkenv is a graphical interface that provides automatic animation able to animate the flow of messages in the network structure and motion of individual objects (Figs 6 and 7). Objects inspection allows to display in the graphical window, state or contents of the object in many ways, e.g. with the use of a chart or manually modify the object. Cmdenv is a plain text user interface [**3, 4, 10**].

3.1 OMNeT++ configuration for wireless sensor network simulations

Simulation of self-reconfiguration protocols used in wireless sensor networks requires the designing of a suitable network node model. The network node located in a tested environment meets the wireless sensor network node task, than we can create simulation scenarios which tests self-reconfiguration techniques. To build a sensor network node we use modules implemented in the INET framework. Wireless sensor network has the ability to generate random events such as disabling a number of nodes, defines transmission parameters in the wireless environment and gives basic addressing of individual nodes in the network. The creation process of simulation is divided into two stages: the first is to create module files in a language *.ned*, the next step is to create the *.ini* file which is a proper simulation file. The OMNeT++ software provides the ability to use both graphical and text interface for work on the *.ned* file (Fig. 8). The modular construction of the source files contributes to the rapid development of new optimized modules. Text mode is characterized by simplicity in code creating, thanks the presenting a syntax in different colours.

The second step of creating simulation is to assign appropriate values for individual parameters corresponding to the module variables previously created in *.ned* files. This process can be performed using the graphical user interface or directly by text edition of individual parameters in the source files. The *.ini* files actually run the simulation. The simulator compiler checks the syntax each time before running the simulation. When running the simulation the program window shows the messages of the modules included in the simulation, the timeline, the current time in the simulation and the number of completed events. Another window contains a graphic representation of a running network simulation where we can see the animation of nodes motion and visualized processes of sending and receiving packets (Fig. 9). When the simulation is completed, the result files are created, where they can be opened and analyzed directly from the simulator. By the use of the graphical user interface with the ability to filter

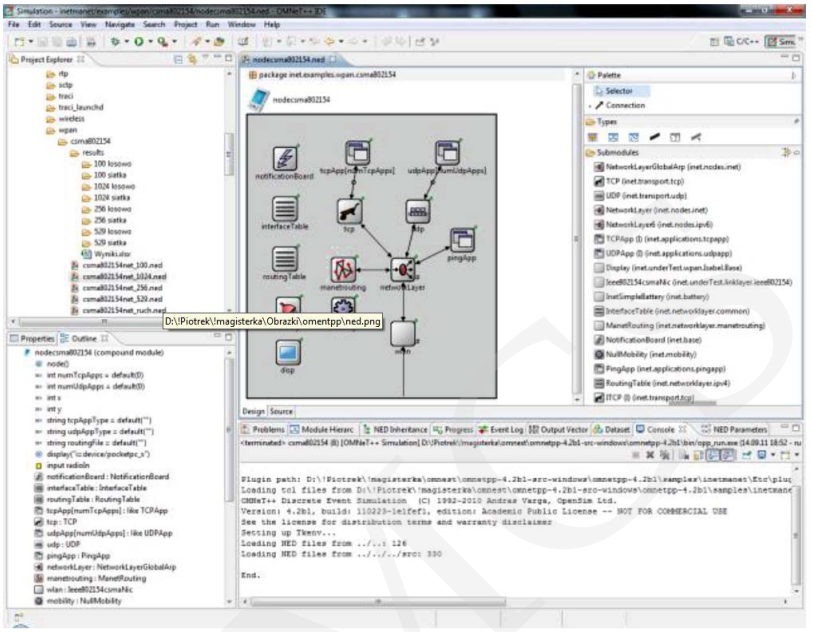

Fig. 8. *.ned* files GUI.

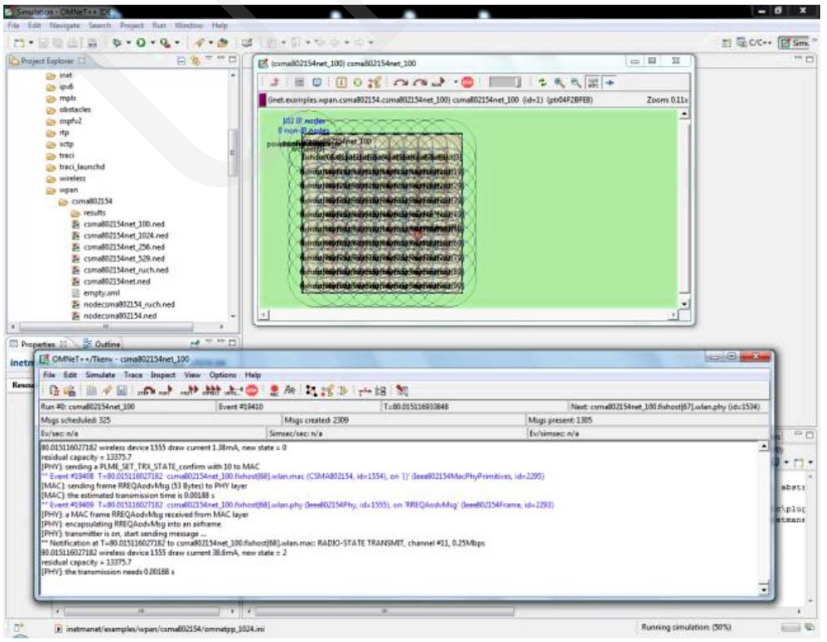

Fig. 9. Screen of the simulation window.

any parameters, we can easily and quickly visualize as graph values measured in the simulation (Fig. 10) [**10**].

3.2 Designing of the sensor network components in OMNeT++

This chapter describes the designing process of a single sensor node model and the entire sensor network in the OMNeT++ environment. These models were used in the study of the self-reconfiguration protocols in the event of a change in the sensor network structure.

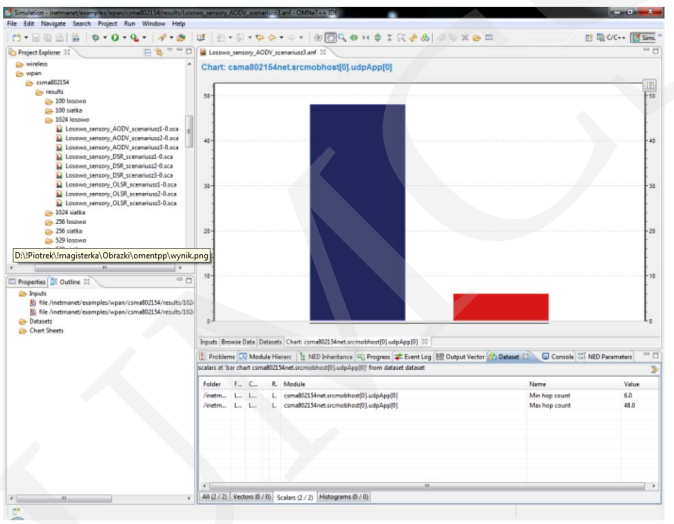

Fig. 10. Stats GUI.

In Fig. 11 a graphical diagram of a wireless sensor network node used in the simulations was shown. It is a graphical model showing the interpretation of a *Nodecsma802154* module in the graphical user interface of OMNeT++ environment. The main component of the wireless sensor network node model is a submodule *IEEE802154csmaNic*. This is the implementation of a wireless network card IEEE 802.15.4, in Fig. 11 it corresponds to the icon with the name *wlan*. The *ManetRouting* module is responsible for choosing the routing protocol, its configuration, and communication through the network layer of the network node model. The *InetSimpleBattery* module is responsible for energy management. Node sends information to the physical layer to indicate the energy level allowing the wireless network adapter operations. The *NotificationBoard* module is used to communicate between modules by sending information about various events such as a transition to upload, routing tables change or low energy level. The *InterfaceTable* module stores information about node interfaces whereas *RoutingTable* stores the routing table for the node. The *NullMobility* module is responsible for node motion programming. Basic node does not have mobility. The *Display* module is responsible for displaying the node communication range and

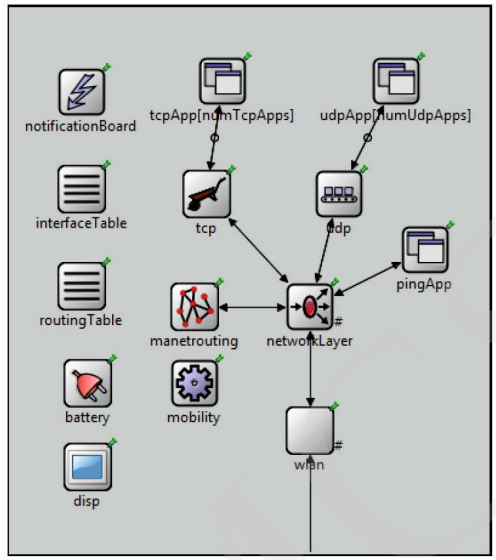

Fig. 11. Wireless sensor network node model in OMNeT++.

setting parameters for communication channel between nodes. The *NetworkLayer* module serves as an interface for transport layer and protocols like TCP and UDP. The *PingApp* class is responsible for the simulation of ping service in Windows OS.

To simulate the wireless sensor network the complete network model was designed with the use of the following modules: *Nodecsma802154* - a node module, described in detail above. *Nodecsma802154_2* a node module which includes the mobility feature. *FlatNetworkConfigurator* module responsible for the addressing individual nodes, which simplifies communication within the network. The *ChannelControl* module is responsible for checking the position of the individual network nodes, its task is to calculate the possibilities for transmission of a particular node taking into account the parameters established by the user. And the last module *PowerControlManager*, a class responsible for generating the failures of individual nodes. Failures result in a lack of ability to send and receive messages from other nodes, regardless of whether the current level of energy allows for it.

4 The study of wireless sensor network self-reconfiguration protocols fault tolerance

This chapter presents the results of research and various simulation runs. In the test the simulation scenario was implemented illustrating the wireless sensor network reconfiguration capabilities in the case of a failure and the ability to provide operations. The key parameter is the number of sensors and the number of failures in the network: 0%, 10%, 25% and 50% nodes failures. The last simulation component is

self-reconfiguration protocol responsible for automatic network routing e.g.: AODV, OLSR and DSR. These protocols are well known in the computer networks, they have a lot of implementations in several programming languages, so that their implementation in the simulation environment is correct and consistent with the specifications of the design such as the ZigBee standard. Furthermore, all of these protocols are often used not only in wireless sensor networks. They are also used in the MANET networks (*Mobile Ad-hoc Network*) and other wireless networks.

The simulation sensor network area is a rectangle with the dimensions of 3x3 kilometers which is covered with 100, 256, 512 and 1024 nodes, where the allocation was carried out with the pre-calculated position, or randomly. The distance between nodes is approximately 80 meters with the case of 1024 nodes. Random arrangement of sensors is faster and less expensive. In the natural environment to deploy sensors a flying aircraft is used. In this model, there are places where the distribution of the nodes is denser and there are the areas where the number of nodes is minimal or empty. For the tests in the simulations a special script was prepared, where the most important parameters were configured as follows: the physical network frequency 2.4 GHz, the signal strength 8mW, 10mW maximum signal strength, base noise 110 dBm, receiver sensitivity -85dBm, the number of communication channels is 11 (capacity 250 kb/s), battery capacity of 2.5 A, voltage 1.5 V and power consumption while sending 9.4 mA, power consumption while receiving is 1.38 mA, power consumption in the standby mode 0.06 mA. The simulation scenario, whose schema is shown in Fig. 12, reflects a motion of a programmed robot (mobile sensor) in the shape of a rectangle (the edge of the sensor field) where sensor moves at a speed of 5 meters per second. Data sent wirelessly through the dynamic route established with the use of the sensors evenly distributed in the area are received by the master sensor, located in the middle of the sensor field. The presented scenario is illustrated in Fig. 12.

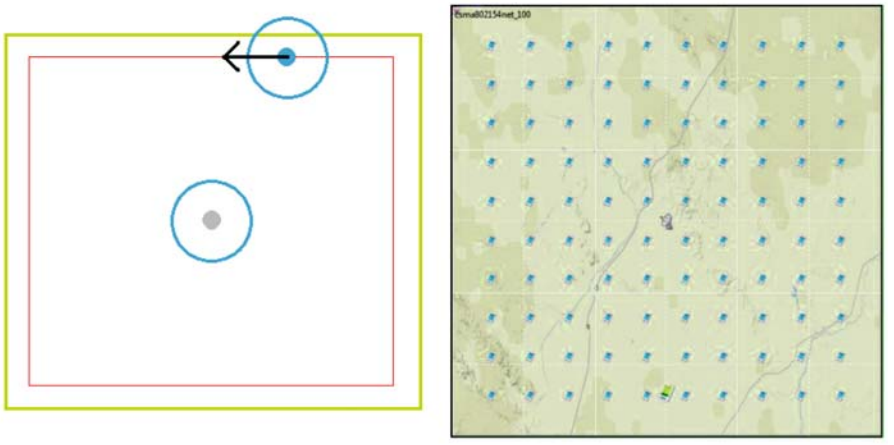

Fig. 12. Simulation schema and sensor nodes allocation pattern.

In Fig. 12 in the center of a sensor area the parent node collecting data from a mobile sensor is located. The program code in Listening 1 allows to implement such a scenario.

Listing 1. OMNeT++ scenario code.

```
*.numSrcHosts = 1
*.numSrcMobHosts = 1

csma802154net_100.srchost[0].x = 1500
csma802154net_100.srchost[0].y = 1500

**.srcmobhost*.mobilityType = "RectangleMobility"
**.srcmobhost*.mobility.x1 = 300
**.srcmobhost*.mobility.y1 = 300
**.srcmobhost*.mobility.x2 = 2700
**.srcmobhost*.mobility.y2 = 2700
**.srcmobhost[0].mobility.startPos = 2.5
**.srcmobhost[0].mobility.speed = 5mps
**.srcmobhost*.mobility.updateInterval = 100ms

**.srcmobhost*.mobility.x = -1
**.srcmobhost*.mobility.y = -1
**.srcmobhost*.x = -1
**.srcmobhost*.y = -1
```

4.1 Self-reconfiguration features simulation results

In order to study the properties of routing protocols to adequately respond to changes in the network topology, the minimal and maximal number of hops in the path from a source node to the destination node was studied. The smaller the number of nodes on the packets route, the faster the data arrives, and fewer nodes energy resources are consumed.

In Figs 13 and 14 we can see that all three protocols respectively coded: AODV - green, OLSR - blue, DSR - red, with full network efficiency have a similar number of hops in all cases. Along with the growing number of node failures in the network, the number of hops increases (axis X in Figs 13 and 14 - the percentage of node failures in the network). For a network consisting of 100 sensor nodes, the difference between the maximal and the minimal number of hops is reduced considerably in the event of a greater percentage of node failures. For the remaining number of sensors in the network, these values remain practically unchanged, only a maximal number of hops is rises. It is easy to notice that the largest increase in the hops number occurs in the DSR protocol. The protocol OLSR has the least number of hops. The protocol AODV has a similar minimal hops value as OLSR. In the case of random distribution of the sensors there can be seen a huge increase in the minimal hops value in correlation to the percentage of node failures. The protocols AODV and DSR in the network distributed randomly over such a large area have problems with the correct assignment of optimal routes. For the protocol OLSR the situation looks better, as the number of nodes in the network increases the situation improves, the hops number is higher than in the

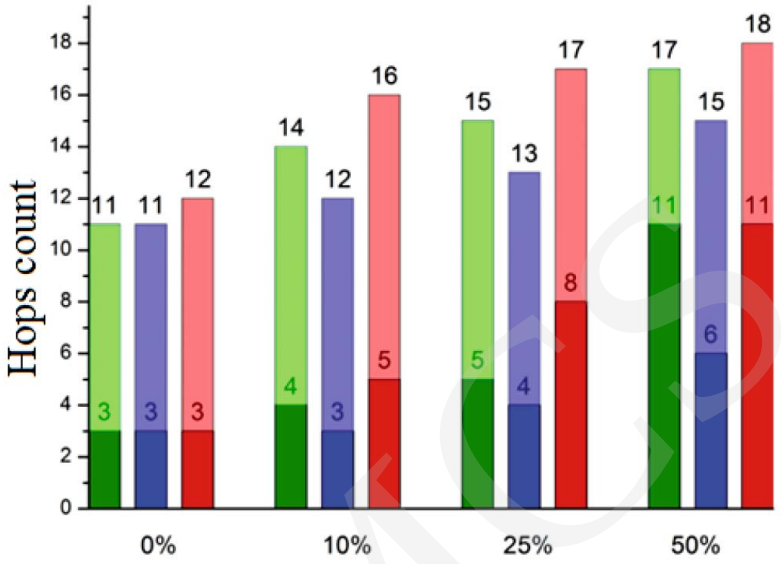

Fig. 13. Minimal and maximal hops count for a sensor field of 100 nodes.

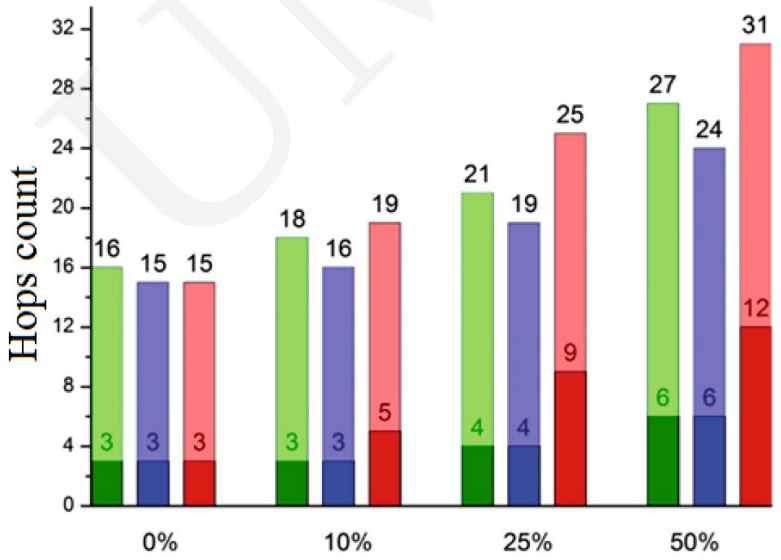

Fig. 14. Minimal and maximal hops count for a sensor field of 256 nodes.

case of uniform distribution. The protocol DSR has the highest maximal values and OLSR the lowest.

An important aspect of the simulation is to study the transmission parameters, which is based on counting packets sent by the source node and comparing them to

the number of packets that arrived to the destination node. There were counted data of transmission control packets and data packets themselves - which made it possible to calculate the ratio of the amount of data sent to the control packets and packets accuracy percent. In addition, the delays on a packets route were measured.

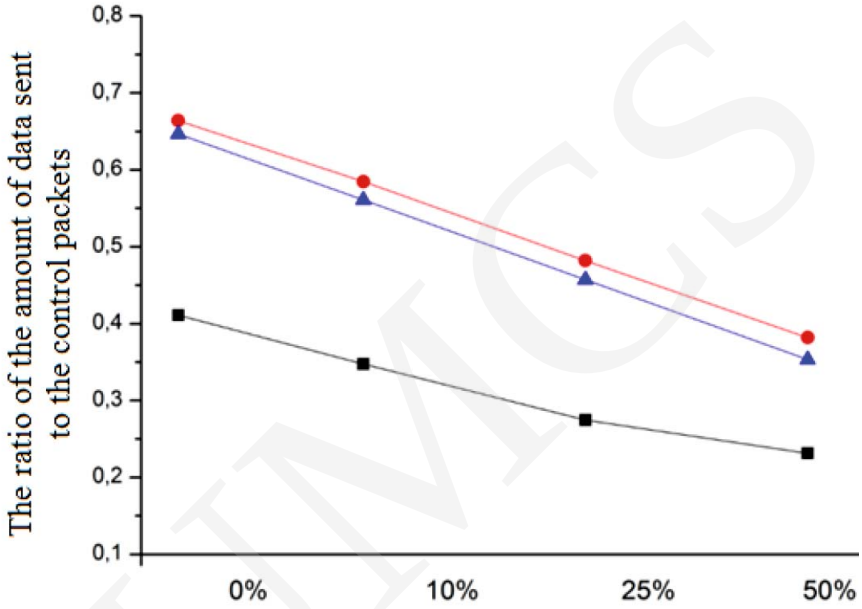

Fig. 15. The ratio of the amount of all data sent to all control packets for 512 nodes.

The ratio of all data packets to the number of auxiliary protocols packets for the mesh of 100 nodes for the protocols OLSR and DSR are negligible, while the AODV protocol significantly differs from the other two. The increase in the network traffic of control packets is due to the increasing ratio of nodes failure in the network. The topologies composed of more nodes, the results of the measurements are shown in Figs 15 and 16 (AODV - black, OLSR - red and DSR – blue line) where decline is slightly smaller, but is still present. Also in such topologies the AODV protocol has the worst performance among the three other tested ones. Using the random distribution for 100 nodes at once, we can see a significant decline in the ratio of sent data packets to the control packets. Also, the decline is much larger than in the uniformly distributed network topology. For 100 randomly distributed nodes and the network failure about 50 percent, the AODV protocol sends control data more than 80% of the overall traffic. By using a larger number of randomly distributed sensors, the network traffic relations improve, but still decrease is greater than using a uniform grid for nodes allocation.

The study of sent packets delay in the scenario is shown in Figs 17 and 18. Respectively, protocols are denoted as: AODV - purple, OLSR - orange, DSR - yellow. The

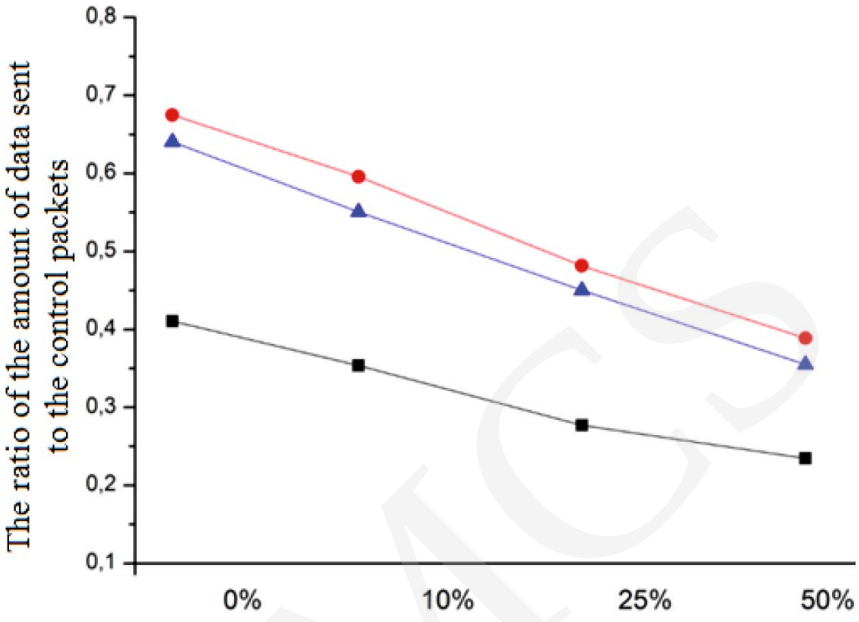

Fig. 16. The ratio of the amount of all data sent to all control packets for 1024 nodes.

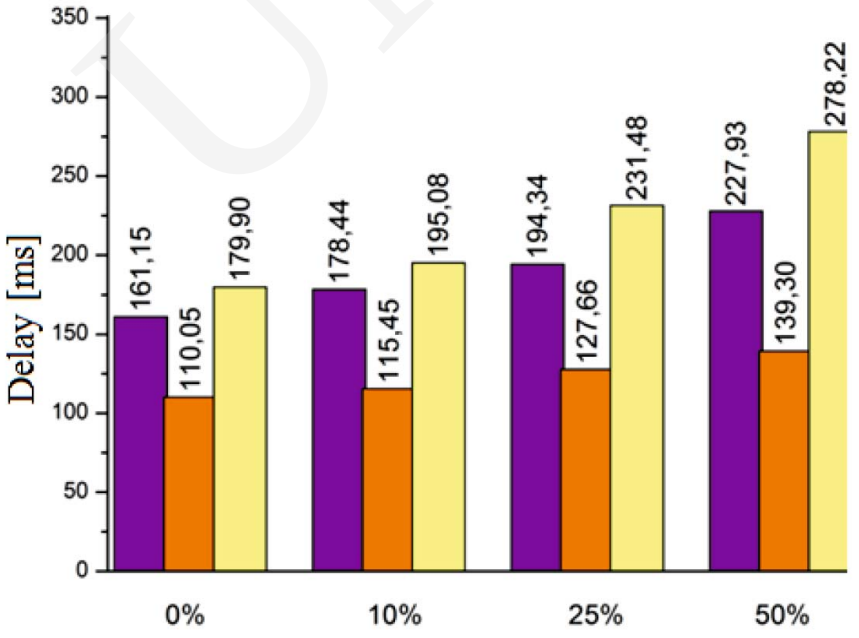

Fig. 17. Packet delay for 512 nodes network.

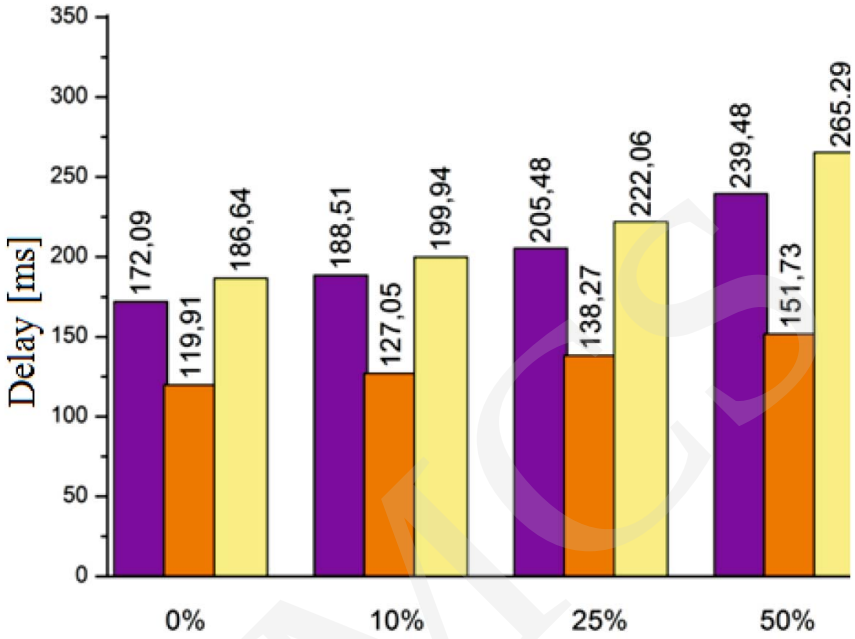

Fig. 18. Packet delay for 1024 nodes network.

smallest transmission delays are found for the OLSR protocol, whether the number of failures on the network increases or not, or if the number of nodes is greater or not. This is due to the small number of hops that packets pass in this protocol. For a grid of 100 nodes the DSR protocol for 50 percent of the nodes failures the delay in sending packets is twice larger than in the OLSR protocol. With the increase of the nodes failures, the delay increases, the smallest increase is in the OLSR protocol. Also the number of nodes in the network has an influence on the increase in packet delay. Based on the carried out simulations which gave the meaningful results we can say that the best self-reconfiguration protocol for the wireless sensor networks was the OLSR protocol. The research shows that the AODV protocol, the number of received data packets dramatically decreases with the increasing number of network nodes failures. The other protocols also significantly lose packets. We can notice that 256 sensors allow proper operation of the network even when the nodes are randomly distributed in the sensor area. An increasing number of sensor nodes significantly improves the percentage of delivered packets.

5 Conclusions

Wireless sensor networks have become a very promising technology that improves everyday life. They present a completely new approach to the topic of measurement

using network solutions. A decentralized network consisting of hundreds or even thousands of autonomous sensors deployed in a defined area, allows remote and non-invasive monitoring of the physical environment parameters. By using specialized communication protocols, this network becomes faults tolerant. It can be a cheap way to implement complex measurement systems, thus improving people's lives in many areas. The wireless sensor network self-reconfiguration is based on the routing protocols – self-reconfiguration algorithms that automatically, without human intervention can manage the network structure to work even if there is severe damage to the structure of the sensor network. Using the free simulation environment which is OMNeT++, provides an easy way to build a sensor network model created with new protocols and introduces new methods of measurement. Increasing importance in the study of wireless sensor networks ensures the quality of service (*Quality of Service*), which can be also present in a simulation environment.

The research carried out on the equipment purchased in the project No POPW.01.03.00-18-012/09 from the Structural Funds, The Development of Eastern Poland Operational Programme co-financed by the European Union, the European Regional Development Fund.

References

[1] Sohraby K., Minoli D., Znati T., Wireless Sensor Networks: Technology, protocols, and applications, Wiley & Sons (2007).
[2] Dymora P., Mazurek M., Nieroda S., Sensor network infrastructure for intelligent building monitoring and management system, Annales UMCS Informatica XII (2012).
[3] Szymanski B. K., Chen G., Sensor Network Component Based Simulator, Handbook of Dynamic System Modeling (2007).
[4] Weingärtner E., vom Lehn H., Wehrle K., A performance comparison of recent network Simulator, IEEE International Conference on Communications (2009).
[5] Peters B., Sensing Without wires: Wireless Sensing Solves Many Problems, But Introduces a Few of Its Own, Machine Design, Penton Media (2005).
[6] Raghavendra C., Sivalingam K., Znati Eds T., Wireless Sensor Networks, Kluwer Academic (2004).
[7] Hajder M., Dymora P., A novel approach to fault tolerant multichannel networks designing problems, Annales UMCS Informatica XI (2011).
[8] Dymora P., Mazurek M., Strzałka D., Long-range dependencies in memory pages reads during man-compute system interaction, Annales UMCS Informatica XII (2012).
[9] Dymora P., Mazurek M., Strzałka D., Statistical mechanics of memory pages reads during man–computer system interaction, Metody Informatyki Stosowanej 1(26) (2011).
[10] Varga A., Using the OMNeT++ discrete event simulation system in education, IEEE Transactions on Education (1999).