**Remigiusz OLEJNIK**

West Pomeranian University of Technology, Szczecin

doi:10.15199/48.2018.06.28

# Network Stack of the HopeMesh Experimental Wireless Mesh Network

***Abstract***. *The article presents an original proposal of a network stack for an experimental wireless mesh network. The stack consists of five layers similar to those of the well-known hybrid approach by Tanenbaum. The physical layer heavily depends on used radio modules; the data link layer uses the Hamming code and the CRC-16 checksum; the network layer is based on the B.A.T.M.A.N. protocol; both transport and application layers are simplified. The paper is concluded with a set of experiments proving the proper behaviour of the entire network stack.*

***Streszczenie.*** *Artykuł przedstawia oryginalną propozycję stosu sieciowego dla eksperymentalnej sieci HopeMesh. Stos składa się z pięciu warstw analogicznych do dobrze znanego hybrydowego podejścia Tanenbauma. Warstwa fizyczna ściśle zależy od użytych modułów radiowych; warstwa łącza danych używa kodowania Hamminga oraz sumy kontrolnej CRC-16; warstwa sieciowa oparta jest o protokół B.A.T.M.A.N.; warstwy: transportowa oraz aplikacji są uproszczone. Artykuł prezentuje ponadto wykonane badania dowodzące poprawności funkcjonowania całego stosu. (**Stos sieciowy eksperymentalnej bezprzewodowej sieci o topologii kratowej HopeMesh**).*

**Keywords:** Wireless Mesh Network, network stack, wireless networks.
**Słowa kluczowe:** sieci bezprzewodowe o topologii kratowej, stos sieciowy, sieci bezprzewodowe.

## 1. Introduction

According to Akyildiz [1], “a Wireless Mesh Network (WMN) consists of mesh routers and clients where mesh routers have minimal mobility and form a mesh of self-configuring, self-healing links among themselves”.

The article presents a network stack that has been designed and implemented for an experimental wireless mesh network. Most of existing WMNs applications depend on standard network devices that are part of the personal computer environment with their operating system. WMNs that are implemented as general-purpose embedded systems are rare.

HopeMesh Experimental Wireless Mesh Network is composed of simple nodes based on a AVR ATmega162 microcontroller with an external 62256 SRAM memory chip that offers additional 32 KiB and a HopeRF RFM12B radio module. The available memory can keep routing data for a maximum number of 2838 nodes, as one entry in the network routing table requires 11 bytes in total.

## 2. An Overview of Existing Network Stack Solutions for Embedded Systems

Currently, there are many available network stacks that can be used as a part of a AVR-based mesh network.

- ***Tuxgraphics*** TCP/IP stack, 3rd generation [2] was designed for distributed sensors with a web server. Its serious disadvantage is the limitation to one IP packet only. Moreover, only the enc28j60 Ethernet module is supported and wireless networking is impossible.
- An open source ***BACnet*** protocol stack for embedded systems [3, 4] consists of an application layer, a network layer, and media access control (MAC) layer communications services for an embedded system or an operating system. There is an existing port for ATmega168 available.
- ***Atmel Lightweight Mesh*** [5, 6] software stack is an easy to use, proprietary, low power wireless mesh network protocol. It has been designed to address the needs of a wide range of wireless connectivity applications, including remote control, alarms and security, automatic meter reading, home and commercial building automation, toys, and educational equipment.
- ***OpenRF***™ is a micro agnostic wireless protocol stack for embedded applications [7]. It comprises three layers, including OpenRF MAC, RadioAPI, and MicroAPI. It was designed in such a way to work with microcontrollers ranging from Microchip's PIC16 series, Atmel AVR and SiLabs 8051 to new Renesas RX62N and RL78 series.
- ***uIP*** (micro IP) and ***lwIP*** [8, 9, 10, 11] were designed by Adam Dunkels from the "Networked Embedded Systems" group at the Swedish Institute of Computer Science. uIP is aimed at tiny 8- and 16-bit microcontrollers and requires very small amounts of code and RAM. It has been ported to several platforms, including DSP platforms. Supported protocols include HTTP, SMTP, FTP, and telnet. lwIP is a stack for embedded systems, used by Altera, Analog Devices, Xilinx, Honeywell and Freescale Semiconductor. Numerous protocols such as PPP, ARP, IP, ICMP, IGMP, UDP, TCP, DNS, SNMP, DHCP are supported by lwIP.
- ***Contiki*** [12] is an operating system designed for networked, memory-constrained systems. It is focused on low-power wireless Internet of Things devices. The system was initially developed by Adam Dunkels in 2002. Further development was done by a worldwide team of developers from Texas Instruments, Atmel, Cisco, ENEA, ETH Zurich, Redwire, RWTH Aachen University, Oxford University, SAP, Sensinode, Swedish Institute of Computer Science, ST Microelectronics, Zolertia, and many others. Contiki provides three network mechanisms: uIP, uIPv6, and Rime stacks. The uIP TCP/IP stack provides IPv4 networking, the uIPv6 provides IPv6 networking, while the Rime stack is a set of custom lightweight networking protocols designed for low-power wireless networks. Contiki can be run on numerous microcontrollers. such as ARM and AVR families from Atmel, dsPIC and PIC32 from Microchip, MSP430, CC2430, CC2538, CC2630 and CC2650 from Texas Instruments, and STM32 from STMicroelectronics.

All of the above-mentioned network stack solutions implement standard protocols, such as IP, TCP, UDP, and HTTP. These protocols are not best suited to Wireless Mesh Network applications. Moreover, these ready network stacks do not support specific hardware. The only solution in such a case is to develop an own lightweight network stack, that will support hardware used along with required higher level protocols.

## 3. Proposal of the Network Stack of the HopeMesh Network

The network stack designed for the HopeMesh network is based on a hybrid network model by Tanenbaum [13] and provides full functionality of five layers. The advantage of the Tanenbaum’s model over the classic OSI model is a

pragmatic approach to the network stack, having less layers than its OSI pendant. This also fits constraints imposed by the embedded architecture used in our HopeMesh implementation.

In this section, we discuss our modifications introduced to the Tanenbaum's network stack along with some implementation details and the software used.

### 3.1. Layer 1 – Physical

The physical layer is implemented using a wireless technology by utilizing capabilities of the HopeRF RFM12B low cost ISM band transceiver. This transceiver offers the frequency-shift keying modulation (FSK) and operates on one of ISM range frequencies – a 868 MHz version has been used in our experiments. The output RF power was set to 4 dBm (≈ 2.5 mW). The antenna used was a simple quarter wavelength wire.

This transceiver can only operate in a half-duplex mode. Thus, a special care has to be taken when writing device drivers for this module. The algorithmic solution to this problem was already presented in another paper [14].

### 3.2. Layer 2a – Medium Access Control

The Medium Access Control (MAC) sublayer was implemented exactly as proposed by Tanenbaum [13] with all his assumptions (Station Model, Single Channel Assumption, Collision Assumption, Continuous Time and Carrier Sense).

The actual MAC frame format is presented in Figure 1. The frame consists of four parts:

- ***Preamble:*** 2 bytes (values `0xAA`) – this pattern is the bit representation of alternating 0 and 1 values; this helps the receiving PLL circuit to lock in to the correct frequency in order to minimize bit errors for the sync pattern recognition.
- ***Sync pattern:*** 2 bytes (value `0x2DD4`) – programmed into the hardware of the RFM12B module; after the module recognizes this pattern, it will fill its internal receiver FIFO with further data.
- ***Data:*** an actual payload for the transmission; the data must not contain data with `0xAA` values.
- ***Postamble:*** 1 byte (value `0xAA`).

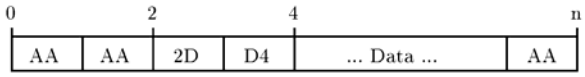


Fig. 1. MAC frame format

One can see that the implemented MAC frame format does not include any error checking. The algorithm will simply stream bytes to or from the upper network layers until the postamble byte `0xAA` is being detected. Only then the MAC layer assumes that the frame reception or transmission is complete.

### 3.3. Layer 2b – Logical Link Control

The implementation follows a Tanenbaum's [13] category of an ***unacknowledged connectionless service***. According to Tanenbaum's data link layer design principles, the following functions were implemented in the LLC layer:

- ***Service interface for the network layer:*** each packet is transmitted in one frame.
- ***Transmission error handling:*** an error correction Hamming code and an error detecting CRC-16 code.
- ***Data flow:*** no flow control algorithms implemented.

#### 3.3.1. Error Correction and Detection

Due to the nature of a wireless connection which is a highly unreliable channel, an error correcting as well as an error detecting code were introduced:

- ***Hamming code:*** this code was used to provide data to the MAC frame. The implemented code is an extended ETS 8,4 Hamming code [15] – so two error bits can be detected and recovered. This code ensures that the final code sent via the physical device does not produce continuous streams of 1 or 0 bits required by the MAC layer.
- ***CRC-16 checksum:*** this code was implemented for the payload provided by the upper network layers. It is used after and before the Hamming decoding process. The algorithm used was reused from the existing AVR `libc` library which provides a hardware optimized CRC-16 routine.

#### 3.3.2. Frame Format

The frame format for the LLC layer can be seen in Figure 2. The maximum size of a frame is 256 bytes, including the LLC header. The following fields were used:

- ***Type:*** 4 bits long – identifies the type of the transmitted packet (broadcast or unicast);
- ***Length:*** 12 bit long – includes the total length of the data payload;
- ***CRC:*** 16 bit long – holds the calculated CRC-16 checksum.

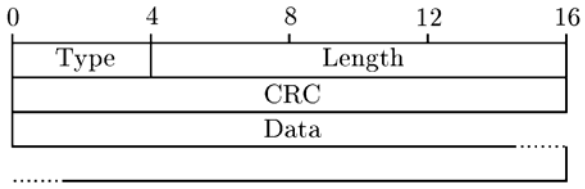


Fig. 2. LLC frame format

### 3.4. Layer 3 – B.A.T.M.A.N. Routing

According to Tanenbaum [13], the network layer is responsible mainly for transparently routing packets from a source to a destination inside the network. The following requirements for this network layer were implemented:

1. **Topology information:** The network layer knows the network topology by using the B.A.T.M.A.N. routing algorithm.
2. **Technology independence:** The network layer completely hides the routing technologies and algorithms from the upper layers.
3. **Uniform addressing:** The upper layers have to provide an uniform address of the destination by using a 16 bit long address value.

The network layer was implemented as a connectionless service which can be qualified as a datagram subnet. Each packet contains the full source and destination address and each router (mesh node) does not hold any state information about connections.

The network layer uses the B.A.T.M.A.N. approach enhanced in the following way:

- ***RFC conformance:*** the real B.A.T.M.A.N. draft RFC [5] was used as a template for implementing the routing algorithm.
- ***TTL introduction:*** to prevent the infinite message looping inside the network.
- ***Bidirectional link check:*** to ensure that only bidirectional routes are being propagated through the network.

#### 3.4.1. Route Propagation

As was already shown in [17] the B.A.T.M.A.N. algorithm is very promising since not the whole network topology information has to be stored on each mesh node but only the next visible gateway for each known destination.

The B.A.T.M.A.N. RFC [16] proposes that each mesh node broadcasts OGM (originator) packets through the network in order to keep the route information up-to-date.

The implemented OGM packet format can be seen in Figure 3. The following fields are being transmitted along with each OGM packet:

- ***Version:*** 4-bits long – specifies the protocol version.
- ***Flags:*** 4-bits long – can carry up to four status bits for the packet.
- ***TTL:*** 8-bits long – decreased before each retransmission. If the value reaches zero, the packet will be dropped.
- ***Sequence Number:*** unique for each OGM and increased with every transmission.
- ***Originator and sender address:*** an address of the node which created and sent the current packet.

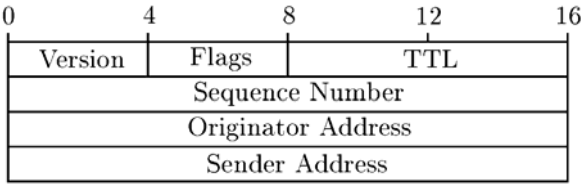


Fig. 3. OGM packet format

Each mesh node always sends an OGM packet after a configurable interval (every second in our case). When a neighbor mesh node receives an OGM packet, it will decide whether to save the originator in the routing table and whether to resend the received OGM packet in order to spread the information about the originator in the network. The full algorithm is shown in Figure 4.

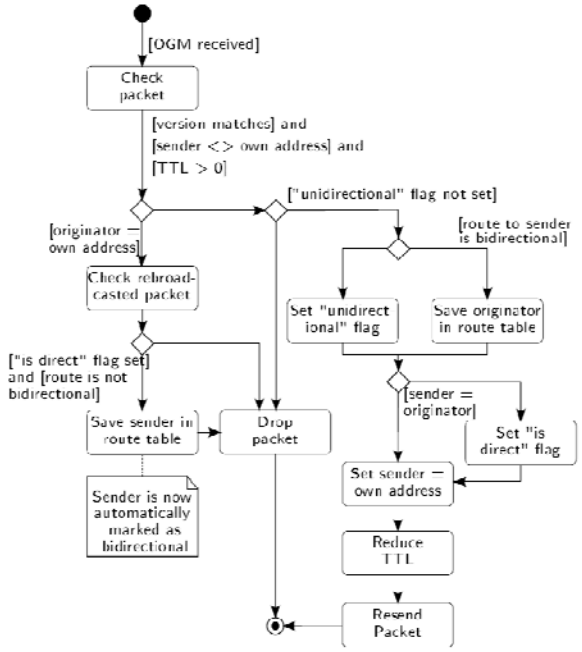


Fig. 4. OGM packet reception algorithm

### 3.4.2. Unicast Messages

Unicast messages carry the actual data payload which has to be routed from an originator to a target. They are fully supported by the HopeMesh implementation, however, they are not described here.

### 3.4.3. Route Determination

The algorithm for finding the best gateway is based on the count of received OGM messages as a metric. The routing table consists of the following fields:

- ***Target address (2 bytes):*** for whom the current routing entry is valid.
- ***Gateway address (2 bytes):*** a possible gateway for the target.
- ***Sequence number (2 bytes):*** the last received sequence number.
- ***Count (2 bytes):*** the total count of received sequence numbers.
- ***Time (2 bytes):*** time of the last update for this routing entry.
- ***Next pointer (1 byte):*** points to the next routing entry.

The routing table with those fields is constantly updated upon the reception of new OGM packets. Every second the routing table is examined for entries which can be purged. The purge timeout is configurable (10 seconds in our case). Route entries older than 10 seconds will be deleted from the routing table. The algorithm ensures that no outdated routing entries are persisted in the routing table. This allows adapting the routing information dynamically with a changing network and conforms to the requirements stated in the RFC [16].

### 3.5. Layer 4 – Transport

Requirements for the transport layer were inherited from the lower layers. A transport layer entity format was chosen as shown in Figure 5. It simply consists of a null-terminated data stream. This allows implementing string based applications in the application layer and passing those strings without further modification to the transport layer.

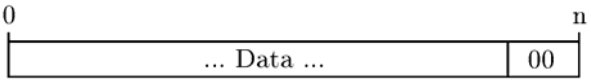


Fig. 5. Transport entity format

### 3.6. Layer 5 – Application

User space applications utilizing the lower network layers currently are:

- ***Shell:*** responsible for sending strings over the network whenever the user prompts a “send” command.
- ***Receiver thread:*** prints received strings to the UART interface whenever the transport layer receives a new payload.

## 4. Experimental Results

The experiments have been conducted to study behavior of the B.A.T.M.A.N. routing algorithm as the heart of the HopeMesh Wireless Mesh Network.

**Experiment #1** This experiment starts with a rebroadcasted OGM sent from `0xc` (a direct bidirectional link neighbor) originated by `0xd`, an unknown node until now (see Figure 6 and Listing 1).

**Listing 1. Output of Experiment #1**

```
rx:
llc: crc=0x0 , len=8, type=1
ogm: sender_addr=0xc, originator_addr=0xd, flags=0x0, seqno=23, ttl=49
tx:
llc: crc=0x0, len=8, type=1
ogm: sender_addr=0xa, originator_addr=0xd, flags=0x0, seqno=23, ttl=48
routing table:
target_addr: 0xb, gateway_addr: 0xb, seqno: 0, cnt: 2, time: 0
target_addr: 0xc, gateway_addr: 0xc, seqno: 0, cnt: 1, time: 0
target_addr: 0xd, gateway_addr: 0xc, seqno: 23, cnt: 1, time: 0
```

As a result, a node `0xd` was added to the routing table. The gateway `0xc` is a known neighbor so a secure route can persist.

**Experiment #2** A goal of the next experiment (performed according to the Figure 7 and Listing 2) was to verify the reception of an OGM originated from `0xc`.

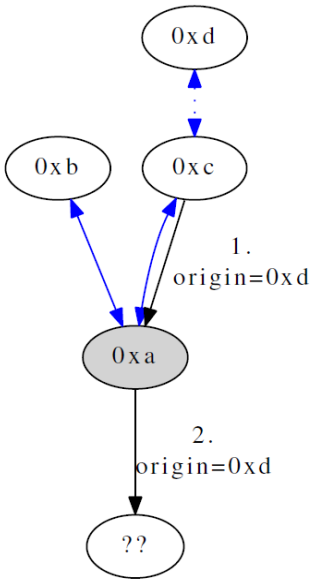


Fig. 6. Route experiment #1

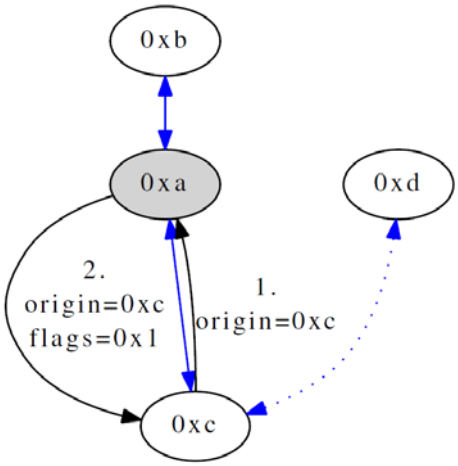


Fig. 7. Route experiment #2

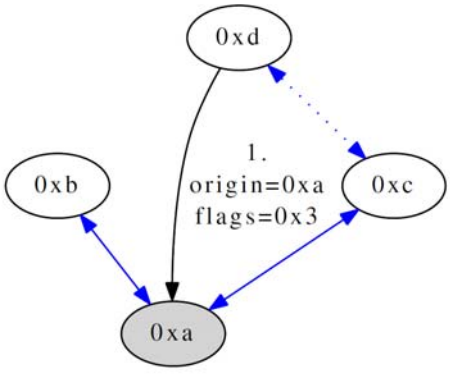


Fig. 8. Route experiment #3

**Listing 2. Output of Experiment #2**

```
rx:
llc: crc=0x0, len=8, type=1
ogm: sender_addr=0xc, originator_addr=0xc, flags=0x0, seqno=1, ttl=50
tx:
llc: crc=0x0, len=8, type=1
ogm: sender_addr=0xa, originator_addr=0xc, flags=0x1, seqno=1, ttl=49
routing table:
target_addr: 0xb, gateway_addr: 0xb, seqno: 0, cnt: 2, time: 0
target_addr: 0xc, gateway_addr: 0xc, seqno: 1, cnt: 2, time: 0
target_addr: 0xd, gateway_addr: 0xc, seqno: 23, cnt: 1, time: 0
```

As expected, the OGM rebroadcasted by `0xa` with the "is direct" flag set.

**Experiment #3** In this experiment a new interesting situation was introduced to the node `0xa`. It suddenly received a direct OGM from 0xd originated by `0xa`. `0xd` was already present in the routing table via the gateway `0xc`. The behavior of the algorithm is shown in Listing 3 and Figure 8.

**Listing 3. Output of Experiment #3**

```
rx:
llc: crc=0x0, len=8, type=1
ogm: sender_addr=0xd, originator_addr=0xa, flags=0x3, seqno=2, ttl=49
routing table:
target_addr: 0xb, gateway_addr: 0xb, seqno: 0, cnt: 4, time: 0
target_addr: 0xc, gateway_addr: 0xc, seqno: 1, cnt: 2, time: 0
target_addr: 0xd, gateway_addr: 0xc, seqno: 23, cnt: 1, time: 0
target_addr: 0xd, gateway_addr: 0xd, seqno: 0, cnt: 1, time: 0
```

The node `0xd` was added as a direct bidirectional link neighbor because it rebroadcasted our own OGM. Note that now we have two route entries to the node `0xd`, one direct link and one via the `0xc` gateway.

**Experiment #4** The final network topology after all the three experiments is shown in Figure 9. The last experiment shows what will happen if no OGM is received after an interval of 16 seconds. This results is the empty routing table which is a correct behavior. Currently, after 10 seconds a route entry must be deleted if no OGM is received within this timeout.

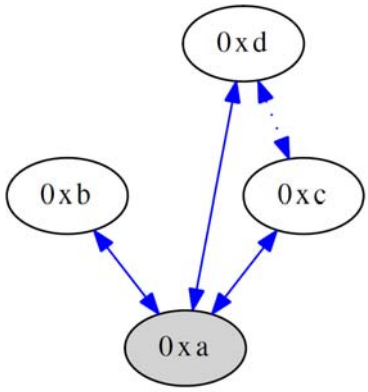


Fig. 9. Final route topology

**Packet drops** The measured percentage of dropped OGM packets was between 0.4 % and 3.97%, depending on the experiment. This is a much better result than a 15%-drop of OMG packets obtained previously in [17].

The research shows that the experiments succeeded in a correct behavior of the routing algorithm implemented according to the proposed network stack. Certainly, not all cases of a network topology update have been studied, which opens new scenarios for future implementations.

## 5. Summary

An implementation of a robust and scalable mesh network using embedded devices despite its hardware constraints is quite possible. A working mesh network prototype and an algorithmic framework for further development is ready and running.

A complete network stack was designed and implemented, as discussed in the paper. The B.A.T.M.A.N.-based routing algorithm was implemented as a network layer protocol. Moreover, some experiments have been carried out to prove the proper routing behavior in dynamic conditions.

Nevertheless, there are some open issues which could be addressed in the future. The following aspects are left open:

- **Node addresses** currently have to be set up manually – this can be done either at compile time or at runtime in the shell.
- There is still significant amount of **packet drops** – the reasons should be investigated.
- Further **network topology** changes and complexities could be studied.
- **Sequence numbers**: currently are mostly ignored for route metrics – this could be also enhanced.
- The current **transport layer** implementation is practically not existent. Further implementation could implement a real layer 4 protocol.
- At the **MAC layer** further enhanced MAC protocols could be investigated.
- **Additional periphery,** such as PS/2 keyboard and a HD44780 LCD display, can be connected instead of the USB terminal connection.

***Acknowledgement***

*I would like to thank my graduate student, Sergiusz Urbaniak, who implemented my preliminary ideas of RFM12B and AVR ATmega16 based wireless mesh network in his master dissertation [18].*

***Author**: dr inż. Remigiusz Olejnik, West Pomeranian University of Technology, Szczecin, Faculty of Computer Science and Information Technology, ul. Żołnierska 49, 71-210 Szczecin, Poland, E-mail: r.olejnik@ieee.org.*

REFERENCES

[1] Akyildiz I. F., Wang X., Wang W.: Wireless Mesh Networks: a Survey. Computer Networks, vol. 47, no. 4, March 2004, pp. 445–487.
[2] Web page: http://tuxgraphics.org/electronics/200905/embedded-tcp-ip-stack.shtml
[3] Web page: http://bacnet.sourceforge.net/
[4] Web page: http://www.bacnet.org/
[5] Web page: http://www.atmel.com/tools/LIGHTWEIGHT_MESH.aspx
[6] Web page: http://www.atmel.com/Images/Atmel-42028-Lightweight-Mesh-Developer-Guide_Application-Note_AVR2130.pdf
[7] Web page: http://openrf.codeplex.com/
[8] Web page: http://en.wikipedia.org/wiki/UIP_(micro_IP)
[9] Web page: http://dunkels.com/adam/mobisys2003.pdf
[10] Web page: http://en.wikipedia.org/wiki/LwIP
[11] S. Zoican, M. Vochin "LwIP stack protocol for embedded sensors network" 2012 9th International Conference on Communications (COMM), 2012, Pages: 221 - 224, DOI: 10.1109/ICComm.2012.6262590
[12] Web page: http://www.sics.se/~adam/contiki/
[13] Tanenbaum A.: Computer networks. Prentice Hall, Upper Saddle River 2002.
[14] Olejnik R.: Modelling of Half-Duplex Radio Access for HopeMesh Experimental WMN Using Petri Nets. Communications in Computer and Information Science, vol. 431, pp. 108–117. Springer International Publishing, Cham 2014.
[15] Datasheet: Enhanced Teletext specification. European Telecommunications Standards Institute, 1997.
[16] Wunderlich S., Lindner M., Aichele C., Neumann A.: RFC draft: Better Approach To Mobile Ad-hoc Networking (B.A.T.M.A.N.). IETF, 2008.
[17] Olejnik R.: An Experimental Wireless Mesh Network Node Based on AVR ATmega16 Microcontroller and RFM12B Radio Module. Communications in Computer and Information Science, vol. 79, pp. 96–105. Springer, Berlin Heidelberg, 2010.
[18] Urbaniak S.: Communication algorithms and principles for a prototype of a wireless mesh network. Master thesis, West Pomeranian University of Technology, Szczecin 2011.