Scientific Journal of Riga Technical University
Computer Science. Technologies of Computer Control
2011
Volume 48

Passive Wireless Sensor Network Analyzing at Medium Access Level

Gundars Miezitis[1], Romans Taranovs[2], [1,2]*Riga Technical University*

***Abstract* – A wireless packet sniffer focused on wireless sensor network analysis, using medium access control protocol packets, has been proposed. Proposed infrastructure and its parts are described . The main goal of this research is to develop simple network infrastructure and program for wireless sensor network analysis which is presented in the following chapters. In this research two main assumptions are made – first, network sniffing is used, which does not disturb network operation, and second, offline data analyzing is used, which means that the data is analyzed when all of it is collected and saved on a computer, but data still is analyzed in real time.**

***Keywords* – medium access layer, passive monitoring, sniffer, wireless sensor network.**

I. Introduction

Wireless sensor networks (WSN) are wireless networked embedded sensing systems allowing monitoring of environment. WSN is composed of many sensor nodes which are limited in resources, such as available energy (they run on batteries), computational power, memory, limited communication range and therefore, it is hard or impossible to perform maintenance of these nodes. That leads to conclusion that nodes must work out energy efficiently – the less energy is consumed during their operation the longer network will be operational. One way to assure it is the development of energy efficient protocols.

Likewise on many network devices, on sensor nodes, too, OSI reference model layers are implemented. Complex WSN can be implemented with five (out of seven) layers – physical layer, data link layer, network layer, transport layer and application layer.It is possible to implement WSN with fewer layers [5]. But usually WSN will be implemented with a data link layer which is responsible for interconnecting two (point-to-point) or more (point-to-multipoint) sensor nodes. And in this layer medium access control (MAC) protocol is implemented. If MAC protocol is well made, network lifetime can increase rapidly. When implementing MAC there is trade-offs like energy efficiency and communication efficiency – with complicated communication schemes it unlikely to be energy efficient and vice versa; if node is in a sleep mode, the data refresh rate decreases; etc.

But how to make sure that implemented MAC protocol is better in comparison. The most common way how to make analysis of a developed protocol is to use some simulation tools, e.g. AVRORA, OmNet++, SENS, TOSSIM, EmTOS, [7] to emulate this protocol in WSN. But simulations lack the influence of real world situations, like radios signal propagation, sensor stimuli and mechanical/chemical stress on sensor node, which cannot be simulated.

Other way how to make analysis of the protocol could be the use of sniffers or packet/protocol analyzer. Sniffer is a program or hardware that can intercept and save packets passing over a digital network or a part of it [7]. In this way it is possible to include in the analysis real world phenomena which could influence protocol efficiency. A drawback of this approach is that you need to collect data in real time whereas in simulation it could be done faster. One of the motivations for developing WSN sniffer was lack of research dedicated to MAC protocol analysis in real life situations. There are researches connected with using sniffers in WSN, which will be discussed in the second chapter, but usually these researches are targeted on different research goals or they lack detailed analysis of MAC level at all.

WSN sniffers can be divided into two types of operation [4]:

- passive;
- active;

Where *passive* monitoring means that network nodes (i.e. task they are executing) are not altered to leak additional information, about node itself, in the network – the WSN is executing its task and nothing else.

Whereas in *active* monitoring node, the task is altered to leak additional data about a node – like remaining energy level; sent/received packet count; etc. – in the network. This means that more packets than needed to execute their task are sent and node energy is depleted more rapidly.

Other way to divide sniffers is by data analyzing type [1]:

- online;
- offline;

Online data analysis is performed in real time, when network is communicating, but *offline* analysis is done later, when certain amount of data is collected.

The goal of this research is to offer a sniffer that could be used for MAC protocol analysis and comparison. This research includes developing program on a sensor node, which collected data and sent it to PC, and program on PC which analyzed the obtained data.

Rest of the paper is organized as follows. In the second chapter related work is discussed. In the third chapter sniffer infrastructure and components are discussed. The following chapter concentrates on criteria used in data analysis. The fifth chapter describes future work and possible ways how to improve the proposed approach. Last chapter concludes this research.

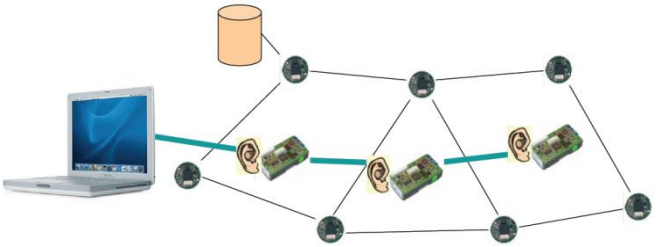

Fig. 1. SNIF architecture [1]

II. Related Work

There are several other WSN analyzing/testing related researches. And in this chapter frequently mentioned related researches will be discussed.

One of the first WSN monitoring systems developed is described in [1] and is called SNIF: Sensor Network Inspection Framework (Fig. 1.). The main idea is to use deployment support network (DSN) which is placed alongside the WSN network during deployment and passively collects packets. It means that a tested network is not affected by SNIF. DSN nodes consist of two RF front-ends: one for listening to WSN nodes – CC1000, other for communication between DSN nodes – Zeevoo ZV 4003 Bluetooth 1.2 located on BTnode. The DNS nodes are time synchronized in between each other to ensure precise packet arrival time.

SNIF can determine communication topology; nodes that have run out of energy; packets route to the sink; if the network has partitions. SNIF is mainly used for application level testing without paying any attention to MAC protocol.

Similarly to [1], [2] uses passive monitoring approach, as well. This system, called Pimoto, consists of three parts: (1) observing node – which collects packets within its range; (2) gateway that retransmits data from observing node to (3) specialized PC server, using TCP/IP. Packets are analyzed and visualized on the server using Wireshark. Like SNIF, Pimoto also uses BTnode with two radios – one for sniffing, other for sending data to PC; data then is sent from PC to a central server where it is analyzed and visualized.

If more than one monitoring node is in vicinity of PC, all monitoring nodes, can send data to one PC.

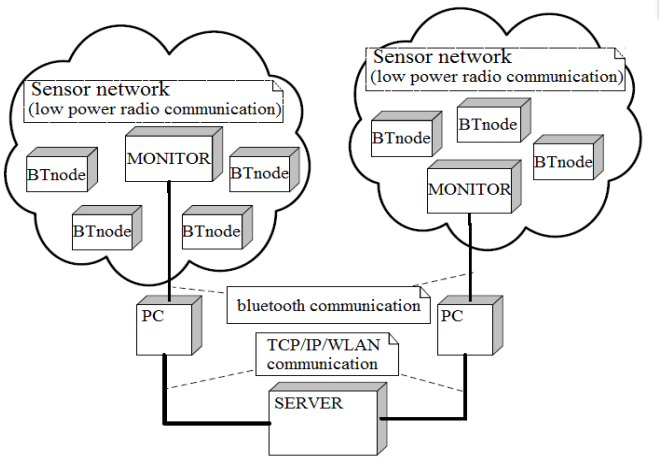

Fig. 3. Pimoto architecture [2]

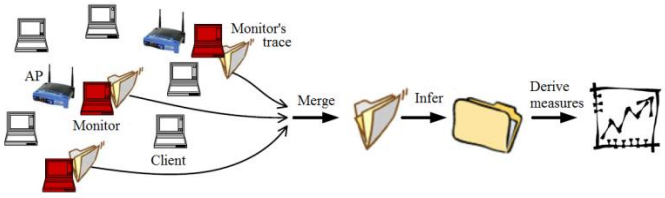

Fig. 2. Wit data analysis pipeline [4]

[4] describes another passive monitoring system – Wit – which has been developed to test 802.11 MAC layer. Wit uses offline approach, merging traces of network traffic collected by distributed sniffers.

Wit uses three steps to analyze data (Fig. 2). In the first step data merging is made; result is the enhanced trace. In the second step formal language recognition tasks are used to conclude which nodes got corresponding packets and which packets have been lost – infer lost packets. In the third step Wit analyzes the enhanced trace and derives different network performance metrics.

LiveNet is a sensor network tool for network dynamics analysis. It uses passive sniffer nodes that forward overheard packets on the serial port to a connected laptop computer or stores them locally in the flash memory. Using an out-of-band mechanism, traces are collected on a central server and merged, based on the Wit approach. LiveNet is used to analyze routing, i.e. it is used in reconstruction of the spanning tree routing paths by means of statistical methods [8].

SNTS: Sensor Network Troubleshooting Suite uses distributed sniffer sensor nodes that record overheard packets in Flash storage. After the experiment, the nodes are collected and the packet traces are transferred to a central server. In contrast to Wit and Jigsaw, where the 802.11 packet format is standardized, SNTS decodes the raw packet dumps based on a text file that describes the packet format. As an example for a possible processing of packet traces, the authors employed machine-learning algorithms to identify bad sequences of events, which lead to an observed bug in the protocol/system, allowing them to fix the problem [7]

Another approach to analyze WSN, as mentioned, is to use an active approach. In this approach sniffer affects node tasks – nodes must periodically send additional information in the environment so that sniffer could capture it and use in analysis.

In Sympathy [3] all sensor nodes periodically send local information to sink node(-s). Sink collects data from three sources – first, each sensor sends additional data of its state, second, is a sink node itself, and third, the area of sink node, because it constantly observes data flow around it.

Sympathy tries to find what caused node failure and localize position of a failed node.

Another *online* system is called Jigsaw. Like Wit, it is used to determine 802.11 MAC metrics. It utilizes time synchronization on the sniffer nodes to reduce the computations costs for merging the traces. Data analysis is made on a single server in real time [11]. Other related

researches include [9], [10]. Short comparison of related work is summarized in Table I.

III. Proposed Passive Monitoring Approach

Passive approach is chosen because when the network is set up (after testing) additional packets will not be needed thus changing network lifetime. But a drawback of this approach is that it is not possible to find out additional information about a node itself.

To improve effectiveness of proposed passive sniffer, simple network infrastructure has been chosen, see Fig. 4 and chapter III-B. This infrastructure provides that all packets are collected.

A. Hardware

As it could be seen in the second chapter, in SNIF and Pimoto used hardware was relatively complex e.g. with two radios. But if someone wanted to adopt this approach they would have to buy similar or exactly the same sensor nodes. Thus, suggestion is to implement a sniffer with simpler hardware on a sensor node, which would decrease testing costs and would allow implementing the proposed approach on a wider range of sensor nodes.

So hardware of choice is ez430-RF25000, which has 16 bit RISC processor MSP430 and radio chip CC2500. Evolution kit can be purchased online – [12] – where even code examples are available. MSP430 maximal data transmission speed is 500 kbps and it operates on 2.4 GHz bandwidth. On board timer is driven by 32768 Hz quartz which implies that packet reception can be measured in ms. It still needs to be tested if that is sufficient for the proposed system.

On computer, programming language Java (*NetBeans* IDE 7.0) is chosen which enables the use of the proposed approach on all operating systems, which have Java Virtual Machine installed on them.

B. Infrastructure

In Fig. 5 it is possible to see proposed infrastructure of network, in which sniffer will be used to test MAC protocol. It consists of four elements: 1) sniffer node; 2) two network nodes; 3) traffic regulator; 4) PC with developed data analyzing program. Proposed network infrastructure has been chosen because it ensures that a sniffer node can hear every packet in the network (assuming nodes are close to each other). In the previously mentioned sniffer systems, like, Wit, Pimoto, it was a problem, how to ensure collection of all (or as much as possible) packets.

Further each node is described by its function i.e. operation mode. In current realization operation mode is chosen by programmer (different ways to choose operation mode are discussed in chapter V).

Network node 1 and 2

MAC protocol operates on network nodes, which are tested. Nodes should execute a program which ensures all types of tasks, i.e. sending off data array, requesting data to the other node, packet sending with different intervals ranging from ms to minutes. These factors insure more detailed data analysis. Network nodes are sending packets to each other. Network between these two nodes is peer-to-peer type, because no other node can request data sending.

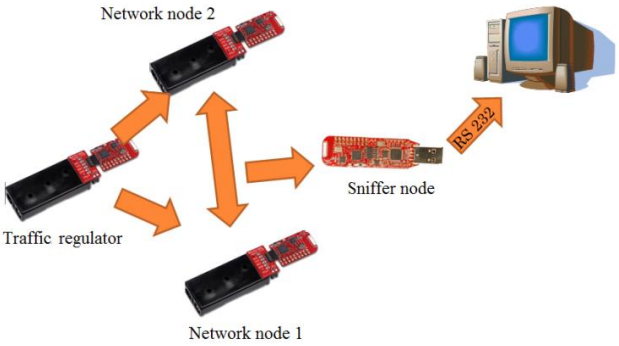

Fig. 4. Infrastructure of proposed system

Nodes do not send information about their internal state, because of passive approach. They send only sensor data to each other.

No other network node is used because MAC protocol, that is connecting two nodes, is being tested.. Common problem in wireless network is hidden node problem. But not always nodes, of proposed infrastructure, are arranged so that hidden nodes exist. This could be tested carefully placing network nodes and traffic regulator node as shown in Fig. 3. Traffic regulator T can send packets to Network node N1, but cannot send packets to network node N2; and similarly wit N2 – N2 can send data to N1, but it cannot hear T. But sniffer node S can hear every node, so no packets are lost for sniffer node. But this would be a special case in testing, because by default there must not be any hidden nodes.

Traffic Regulator Node

This node ensures sending extra packets to the environment, thus disrupting network node communication. This way MAC protocol must work around busy environment to get access to it and send data. By collecting packets, analyzing program is able to conclude MAC protocol criteria. Extra packet sending to the environment happens randomly, thus, there are possible different duty traffics.

Traffic regulator node is also implemented with some simple MAC protocol (it does not have to be the same as the tested MAC protocol). This must be done not to damage current data transaction between network nodes. If this had not been done, MAC metrics would drop rapidly, incorrectly implying bad MAC implementation.

Hidden node can be determined if there are simultaneous (or close) packets sent from N2 and T and if packet collision is detected. Then it is possible to assume that there is a hidden node.

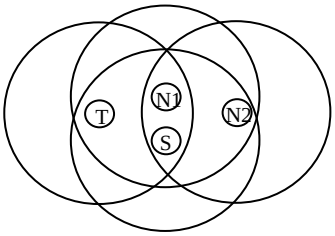

Fig. 5. Testing hidden node problems

TABLE I

COMPARISON OF MONITORING/ANALYZING APPROACHES

Approach	Type of operation	Type of data analyzing	Goal of approach	Pros	Cons
SNIF	Passive	Online	WSN packet routes, communication topology, failed nodes during deployment	Two radio front-ends ensure *online* data analyzing; DSN can be removed and implanted in tested network at will;	Complex hardware is needed; DSN must be set up and disassembled; Like in all passive approaches it is not possible to get additional information of the node state;
Pimoto	Passive	Offline	WSN protocol debugging during their development	Doesn't affect node tasks; Data analysis is conducted as plugin for Wireshark;	If the network under test is large there are needed many PCs and a server; Complex data acquisition scheme; Data analysis is limited to what Wireshark allows;
Wit	Passive	Offline	IEEE 802.11 MAC analysis	Merges distributed traces in one trace; Allows to infer which packets have been lost;	Uses complex trace merging mechanism; Complex inferring mechanism;
LiveNET	Passive	Offline	WSN dynamics analysis	Doesn't affect node operation;	Use centralized server for data analysis;
SNTS	Passive	Offline	WSN automated failure diagnosis	Packet collection is limited with onboard memory; Raw packet data can be viewed;	Must have separate text file that describes used MAC protocol;
Sympathy	Passive /Active	Online	Finding depleted or failed nodes in WSN	Possible to get information about sensor node;	Centralized data collection creates extra data in communication network and increases energy consumption;
Jigsaw	Active	Online	IEEE 802.11 MAC analysis	It is possible to constantly follow how network works in real time – watch reaction on some event, like, placing obstacle in network;	Complex and time demanding hardware to ensure online data analysis;
Suggested approach	Passive	Offline	WSN analysis at MAC level	WSN characteristics can be analyzed with different MAC protocols; MAC packets for network analysis are used; Real time analysis of data; Collection of all packets;	Isn't used in real sensor network, but in proposed infrastructure; Type of used MAC protocol should be known;

Sniffer Node

Basically this node listens to environment until it hears any packet. When it hears a packet it then saves the packet in memory. Memory is organized like two buffers i.e. two arrays of data structure. Data is saved in the first buffer until it is 60% full. When that happens packet saving is switched to the second buffer, but the first buffer content via RS232 interface is sent to PC and stored there. When the second buffer is 60% full data saving switches to the first buffer, but the second buffer content is sent to PC. The 60% limit ensures that in heavy duty traffic extra space is left in memory to save packets. Afterwards buffers are emptied in low duty traffic. Sniffer node is constantly connected to PC so it does not ' need a battery.

When saving a packet on sniffer node, a few additional fields are added to packet content , like time – t_i, i∈[1,n], where t_i – time when *i-th* packet was captured and *n* – all packet count, showing the time when packet was captured (time is local on the node and it starts from 0), radio signal strength indicator – RSSI [*dBm*], link quality indicator which could be useful in further data analyzing. In data structure is left six byte reserve for future improvements.

Data Analyzing Program on PC

Data is saved on PC in one file that can be opened with analysis program. When saving it on PC, packets are saved in a sequential manner – they are sorted by time starting with 0 and ending with t_n. Program will analyze collected data i.e. packets and additional information. Criteria of interest are as follows:

- Mean time to execute task, e.g. sending data array from one node to another - $\bar{t}_{task}$;
- Mean reaction time, e.g. time it takes to respond to other node request – $\bar{t}_{react}$;
- Collision count versus all packet count – n_{coll};
- Repeated packet count - n_{rep};
- Hidden node;
- RSSI [*dBm*];
- LQI;
- etc.

These criteria in a more detailed way are discussed in chapter IV.

C. Testing

To ensure correct data collection and analysis, MAC testing must go through three phases: (1) network infrastructure set-up phase; (2) data collection phase; and (3) data analysis phase.

Network Set-Up Phase

In this phase physical network is placed in the environment. The environment can be either indoors or outdoors, or both (data after analyzing can be compared). Also a user must choose either to test default network infrastructure or to test hidden node problem, described previously.

This phase also includes choosing node set-up, e.g. choosing radio modulation, channel, choosing node operation mode (sniffer, network or regulator node). All the actions and decisions are made by the user/programmer, i.e. user/programmer chooses which node fulfills definite functions.

Data Collection Phase

In this phase actual network operation happens. While network is operational, packets are collected, saved and sent to a computer. The more packets will be captured, the more precise data analysis will be made. It is hard to define minimum packet count, because no researches imply how much packets is enough, so minimum packet count should be determined empirically, during tests. When saving packets on PC every packet is with unique sequence number.

Data Analysis Phase

In this phase the saved data is analyzed and visualized (graphics or tables) on a computer. It basically is working with analyzing program. In this phase data is analyzed by the criteria discussed in the next chapter. Unfortunately analysis program have not been finished yet, so no screenshots cannot be provided.

IV. Criteria

Criteria can be divided by used MAC type in comunication. Generally there can be three types of MAC protocols: 1) demand assignment, 2) fixed assignment, 3) random access protocols [13]

In demand assignment protocols, the exclusive allocation of resources to nodes is made on a short-term basis, typically the duration of a data burst. For example IEEE 802.15.4, Low-energy Adaptive Clustering Hierarchy - LEACH, (centralized protocols) and Token Ring (distributed protocol).

In fixed assignment protocols, the available resources are divided between the nodes so that the node assignment is long term and each node can use its resources exclusively without the risk of collision. For example, time division multiple access - TDMA, frequency division multiple access FDMA and code division multiple access - CDMA.

In random access protocols, the nodes are uncoordinated, and the protocols operate in a fully distributed manner. They often incorporate a random element, for example, exploiting random packet arrival times and setting timers to random values. For example, ALOHA, slotted-ALOHA, carrier sense multiple access - CSMA.

A. Criterion for Demand Assignment Protocols

Protocol Overhead

Protocol overhead is induced by MAC-related control frames, for example request to send – RTS and clear to send – CTS or data request packets in demand assignment protocols.

It is possible to calculate protocol overhead if RTS, CTS, ACK or other control packets are counted:

$$n_{prot\%} = n_{prot}/n * 100\%, \qquad (1)$$

where n_{prot} – overhead packet count, n – total packet count. This criterion can be used in random access protocols too.

B. Criterion for Fixed Assignment Protocols

Synchronization Time Overhead

TDMA based protocols must use time synchronization mechanism to sustain collision free environment. But this creates overhead in traffic.

It is possible to search packets that are connected to node synchronization:

$$n_{sync\%} = n_{sync}/n * 100\%, \qquad (2)$$

Where n_{sync} –packet count that is connected to synchronization, n – total packet count

C. Criteria for All Types of MAC Protocols

Mean Time to Execute Task

When testing mean time of task execution, as task sending sensor data arrays from one network node to another, has been chosen. Sensor data array size can be constant or variable one. When analyzing data, packets that request sensor data array transmission are searched.

Packet, that requests sensor data array, with time: $ttask\_start = t_i$, $i = ntask\_start$,and packet, that sends last sensor data array, with time: $ttask\_end = t_k$, $k = ntask\_end$, are searched within data, i.e. in saved packets from sniffer node; n_{task} – total packet count that requests sensor data array, n – total packet count; t_{task_start} – is time of packet, that requested sensor data array, n_{task_start}– is sequence number of packet that requests sensor data array,t_{task_end} – is time of packet which sends last value of sensor data array, n_{task_end}– is sequence numbers of packet that sends last sensor data array value; after finding all these times it is possible to subtract them:

$$\bar{t}_{task} = \left(\sum_{j=1}^{n_{task}} (t_{task\_end} - t_{task\_start})\right)/n_{task} \ , \qquad (3)$$

Task start and end times are taken from packets that requests array.

For example, sensor data array request packet is found, that has sequence number 102 and record time – 123 (ms). Later last sensor data array packet is found with sequence number 117 and record time 173 (ms). And this is the third pair of sensor data array start and end times. Then using mentioned formula it is possible to write: $t_{task\_start} = 123, i = 102$ and $t_{task\_end} = 173$, k = 117.

Minimal and maximal times to send data arrays can be calculated, too. All the results are saved and put into the TABLE I

Mean Reaction Time

When one node requests data from another node, it is needed to know how long it takes to prepare a reply. This describes node and protocol speed on the node.

Packet, that requests data, with time $t_{req\ start} = t_i, i = n_{req\ start}$ and packet, that responds to that request, with time $t_{req\ answ} = t_k, k = n_{req\ answ}$, are searched within data; n_{req} – packet count that requests data form second node, n – total packet count, where $t_{req\ start}$ – is time of packet, that requested sensor data, $n_{req\ start}$ – is sequence number of packet that requests sensor data, $t_{req\ answ}$ – is time of packet which responds to request, $n_{req\ answ}$ – is sequence numbers of packets that respond to a request.

And now mean value can be calculated:

$$\bar{t}_{req} = \left(\sum_{j=1}^{n_{req}} (t_{req\_answ} - t_{req\_start})\right)/n_{req}, \quad (4)$$

Collision Count Versus All Packet Count

Collision happens when two packets are sent simultaneously from two nodes. In the proposed infrastructure two type collisions can happen: when both network nodes send packets simultaneously and when one of the network nodes sends a packet simultaneously with a packet regulator. For example, collision has happened when acquired data packet has incorrect CRC. But there is one problem – packets can be corrupted not only by collisions, but by environment noise too. This cannot be avoided in passive approach.

Data is scanned counting unusable packets, afterwards this number is divided with all packet count:

$$n_{coll\%}=n_{coll}/n*100\%, \quad (5)$$

where n_{coll} – collision packet count, n – total packet count.

Repeated Packet Count

This criterion shows how much packets have been lost during network operation. With great packet repetition count network operation slows down.

Two identical data packets are searched in input data. Several succesive packets will be compared, because if packets repeat they will be close to each other.

$$n_{rep\%}=n_{rep}/n*100\%, \quad (6)$$

where n_{rep} – repeated packet count, n – total packet count.

Radio Signal Strength Indicator – RSSI

This criterion is acquired when a packet is received by a sniffer node. RSSI is an estimate of the signal level in the chosen radio channel. This value is based on the current gain setting in the RX chain and the measured signal level in the channel. RSSI can be read from dedicated status register [6].

RSSI can describe how far a network node and traffic regulator are from a sniffer node. This could be used to estimate radio output power setting (if all node locations are known). RSSI is measured in *dBm*, which is abbreviation for the absolute power ratio in decibels (dB) of the measured power referenced to one milliwatt (mW).

Link Quality Indicator

It describes how easy it is to take data from radio signal i.e. demodulation. The LQI gives an estimate of how easily a received signal can be demodulated by accumulating the magnitude of the error between ideal constellations and the received signal. LQI is mostly used as a relative measurement of the link quality, since the value is dependent on the modulation format. A higher value indicates a better link than what a low value does. This value can be read from dedicated status register 0.

LQI can show how noisy the environment is – the harder to demodulate packet, the noisier environment is. Unit of measurement is relative – it describes error vs. ideal signal.

V. Future Work

First of all, for programming eZ430-RF2500 nodes are chosen and analyzing program have not been finished yet. Secondly, the developed system must be tested and checked if the proposed assumptions are correct. On the nodes different MAC protocols must be implemented (and no MAC protocol) and results must be compared. After analyzing data, a decision must be made either to improve the proposed approach or to leave it as it is.

There can be other possible ways for improvement of the proposed approach. Semi-active approach could be implemented when available node energy levels are sent to sniffer node at the beginning and end of the protocol testing. Thus information of energy consumption during node operation with minimal packet overload and energy consumption – could be gained. Another way how to improve proposed system is to add centralized management i.e. sniffer node sends commands to other nodes cotrolling testing nodesand their functions . For example, in this case traffic regulator packet transmission intensity could be chosen (constant, random, none). Traffic regulator node could be programmed remotely in setup phase. This means that the first phase of testing should be changed.

The first phase now would be managed not manually by a user, but automatically using a sniffer node and a computer program as a manager. In the computer program, a user should be able to set up node operation modes and functions like modulation, radio channel, etc. Additionally, in this phase, all nodes should inform sniffer about available nodeenergy .

Another way to develop our research would be to create a test bed in which chosen network infrastructure could be evolved.

The Second phase should also be changed– all nodes again must inform sniffer node about their available energy.

VI. Conclusions

Medium access (MAC) protocol is one of the most common protocols implemented onto sensor node. This protocol can increase or decrease lifetime of network itself, because MAC protocol can be energy consuming. But energy consumption may be increased due to the excessive packet retransmission. But it may cause packet collisions, environmental disruptions, bad implementation of protocol, etc. That is why a way to test the developed MAC protocol is needed. The first step to test MAC protocol would be using simulations, but in simulations there cannot be predicted such environmental phenomenon, like radio signal propagation, sensor stimuli and mechanical/chemical stress on sensor node.

So different approach is suggested to test MAC protocols in real life network and analyze data using sniffer. Sniffer, which

is composed of sniffer node and analyzing program on PC, listens to environment and saves every heard packet in one of two buffers, in sniffer memory. The packets are sent to PC over RS232 interface and saved in one file which is later used in data analysis.

Passive data collection is chosen, because it does not disrupt network operation. And offline data analyzing is chosen, because, before analyzing, enough data must be collected.

REFERENCES

[1] Matthias Ringwald, Kay Romer, Andrea Vitaletti, "SNIF: Sensor Network Inspection Framework", pp. 14, Zurich, Switzerland, 2006
[2] Abdalkarim Awad, Rodrigo Nebel, Reinhard German and Falko Dressler, "On the Need for Passive Monitoring in Sensor Networks", Germany, pp. 7, 2008
[3] Nithya Ramanathan, Kevin Chang, "SYMPATHY for the Sensor Network Debugger", SenSys '05, Nowember 2-4, 2005, San Diego, California, USA, 2005
[4] Ratul Mahajan, Maya Rodrigo, David Wetherall and John Zahorjan, "Analyzing the MAC level Behavior of Wireless Networks in the Wild", SIGCOMM '06,September 11-15, Pisa, Italy, 2006
[5] Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci, „A Survey on Sensor Networks", Georgia Institute of Technology, p-13, 2002
[6] Texas Instruments, "CC2500", Texas Instruments, 2009, Available: http://www.ti.com/lit/ds/symlink/cc2500.pdf [Accessed: Aug. 05,2011].
[7] Matthias Ringwald, PhD thesis: "Reducing uncertainty in wireless sensor networks", Zurih Germany, p-189, 2009
[8] Bor rong Chen, Geoffrey Peterson, Geoff Mainland, and Matt Welsh. Livenet: Using passive monitoring to reconstruct sensor network dynamics. In Proceedings of the 4th IEEE/ACM International Conference on Distributed Computing in Sensor Systems (DCOSS 2008), pages 79–98, Santorini Island, Greece, June 2008
[9] V. Handziski, A. Kopke, A. Willig, and A. Wolisz, "TWIST: A Scalable and Reconfigurable Testbed for Wireless Indoor Experiments with Sensor Networks," in 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing (ACM Mobihoc 2006): 2nd ACM International Workshop on Multi-hop Ad Hoc Networks: from theory to reality 2006 (ACM REALMAN 2006), Florence, Italy, May 2006, pp. 63–70.
[10] H. Ritter, R. Winter, and J. Schiller, "A Partition Detection System for Mobile Ad-Hoc Networks," in First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON 2004), Santa Clara, California, October 2004, pp. 489–497.
[11] Yu-Chung Cheng, John Bellardo, Péter Benkö, Alex C. Snoeren, Geoffrey M. Voelker, and Stefan Savage. Jigsaw: Solving the puzzle of enterprise 802.11 analysis. In Proceedings of the ACM SIGCOMM 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Pisa, Italy, September 2006.
[12] Texas Instruments, eZ430-RF2500, 2009 [Online]. Available: http://www.ti.com/tool/ez430-rf2500 [Accessed: Aug. 01, 2011]
[13] Holger Karl and Andreas Willig. "Protocols and Architectures for Wireless Sensor Network", England: John Wiley & Sons, 2005. – 507 p

Gundars Miezitis was born in 1987. Bachelor degree in computer science (Riga Technical University, 2010). Now he is a master student at Riga Technical University, Institute of Computer Control, Automation and Computer Science, Department of Computer Networks and System Technology.
Currently he is working at Riga Technical University. Previous work includes research connected with Wireless Sensor Network localization problems and localization using radio signal strength.
Email: gundars.miezitis@rtu.lv

Romans Taranovs. received M.sc.eng. degree from Riga Technical University in 2009. He currently is a doctoral student at Riga Technical University. He has been holding the assistant position at Riga Technical University since 2009.. His research interests include microprocessors, embedded systems wireless sensors networks and computer based control.
Email: romans.taranovs@rtu.lv

Gundars Miezītis, Romāns Taranovs. Pasīva bezvadu sensoru tīkla analizēšana vides piekļuves slānī.
Pētījuma mērķis bija izstrādāt paņēmienu ar kura palīdzību iespējams veikt bezvadu sensoru tīkla vides piekļuves slāņa analīzi reālā vidē. Lai sasniegtu mērķi tika izpētīti esošie paņēmieni bezvadu sensoru tīklos izmantotajos datu analīzes paņēmienos un sistēmās. Izpēte noveda pie slēdziena, ka eksistē divu veidu analīzes sistēmas – pasīvā, kas neietekmē tīkla darbību, un aktīvā, kas ietekmē tīkla darbību nosūtot vidē vairāk ziņojumus kā nepieciešams. Kā rezultātā tika pieņemts lēmums izmantot pasīvo paņēmienu. Tā realizācijai ir nepieciešams izmantot tā saucamo okšķeri (angl. *sniffer*), kas klausās vidi un saglabā visus uztvertos ziņojumus vispirms uz sensora mezgla esošajā atmiņa, vēlāk datora atmiņā, kur datus nogādā izmantojot RS232 interfeisu. Tika izvēlēta tāda tīkla infrastruktūra, kurā darbojas divi tīkla mezgli, uz kuriem atrodas testējamais vides piekļuves protokols, trafika regulatora mezgls, kas sūta traucējošos ziņojumus tīkla mezgliem un okšķera mezgla, kas savienots ar datoru. Tīkla mezgliem ir jāveic tāda programma, lai varētu izpētīt visus izvirzītos kritērijus – tai ir jāpieprasa/jānosūta datu masīvs vai vienkārši sensoru dati, jāsūta ziņojumi ar dažādām intensitātēm (intervālos no milisekundēm vai mikrosekundēm līdz pat vairākām minūtēm). Trafika regulators paketes vidē nosūta ar gadījuma intervāliem, tādējādi nodrošinot atšķirīgus tīkla noslogojumus. Okšķeris saglabātajām paketēm pievieno tādus papildus laukus, kā laiku, kad pakete dzirdēta, radio signāla stipruma indikatoru un saites kvalitātes indikatoru. Kad visi dati saglabāti uz datora tos analizē izveidotā programma pēc vairākiem kritērijiem – vidējais laiks uzdevuma veikšanai, vidējais reakcijas laiks, kolīziju daudzumu, atkārtoto pakešu daudzumu un meklē vai ir slēpto mezglu problēma. Pasīvās pieejas trūkums ir tāds, ka nav iespējams iegūt nekādu papildus informāciju par mezgla iekšējo stāvokli, bet priekšrocība ir tāda, ka tā neietekmē darbojošos tīklu.

Гундарс Миезитис, Роман Таранов. Пассивное анализирование беспроводных сенсорных сетей на уровне доступа к среде.
Цель статьи заключается в создании техники, с помощью которой можно проводить анализ беспроводных сенсорних сетей на уровне доступа к среде, в реальной среде. Для достижения цели были исследованы существующие подходы и методы анализа данных беспроводных сенсорных сетей. Исследования показали, что есть два типа систем анализа - пассивные, которые не влияют на производительность сети, и активные, которые влияют на работу сетей при отправике дополнительных сообщений, характеризующих сенсорный узел. В результате было принято решение использовать пассивный метод. Для реализации цели было необходимо использовать так называемый „доносчик" или *Sniffer*, который слушает окружающию среду и сохраняет все полученные сообщения сначала в памяти сенсорного узла, потом в памяти компьютера, где данные передаются через интерфейс RS232. Была выбранна сетевая инфраструктура, состоящая из двух узлов сети, на которых находится тестовой протокол доступа к среде, узел-регулировщик, который посылает дополнительные сообщения, и вмешивают комуникацию сетевых узлов, и узел-доносчик, который подключен к компьютеру. Узлы сети должны выполнить программу, которая позволит изучить все критерии, - она должна получать / отправлять массив данных, или просто данных датчика, отправлять сообщения с различной интенсивностью (с интервалом от микросекунды или миллисекунды до нескольких минут). Узел-регулировщик - пакет в среде отправляют со случайными интервалами, обеспечивая различные нагрузки сети. Сохранение пакета обеспечивается дополнительными полями, например – время, когда пакет можно слышать, индикатор мощности радиосигнала и индикатор качества связи. Когда все данные, хранящиеся на компьютере, проанализированы на созданой программе по нескольким критериям - среднее время для выполнения задачи, среднее время отклика, количество конфликтов, количество повторных пакет и ищете нет ли проблема скрытого узла. Недостатком пассивного метода является то, что невозможно получить дополнительную информацию о внутреннем состоянии узла, но преимущество в том, что он не влияет на операционную сеть.