

Czech Technical University in Prague

Faculty of Electrical Engineering

Department of Control Engineering

ITEM, TinyOS module for sensor networks – Testing and application

Diploma Thesis

Author: Bc. Pavel Beneš

Supervisor: Ing. Jiří Trdlička

Thesis Due: May 2010

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Pavel Beneš**

Studijní program: Elektrotechnika a informatika (magisterský), strukturovaný
Obor: Kybernetika a měření, blok KM1 - Řídicí technika

Název tématu: **ITEM, TinyOS modul pro senzorové sítě - Testování a aplikace**

Pokyny pro vypracování:


1. Navrhněte a implementujte aplikaci pro měření a komunikaci dat v senzorové síti s použitím modulu ITEM.
2. Navrhněte a implementujte aplikaci pro testování modulu ITEM.
3. Proveďte testování modulu, zpracujte a vyhodnoťte výsledky.
4. Z naměřených dat vytvořte vizualizaci funkce modulu ITEM.
5. Identifikujte nedostatky modulu ITEM a část z nich odstraňte.

Seznam odborné literatury:

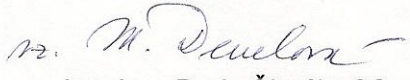
1. Akimitsu Kanzaki, Takahiro Hara, Shojiro Nishio. An adaptive TDMA slot assignment protocol in ad hoc sensor networks. Proceedings of the 2005 ACM symposium on Applied computing, March 13-17, 2005, Santa Fe, New Mexico
2. TinyOS web pages: <http://www.tinyos.net/>
3. Inderjit Singh. Integrated TDMA E-ASAP Module (ITEM). FEL-CVUT, Prague, 2007.

Vedoucí: Ing. Jiří Trdlička

Platnost zadání: do konce letního semestru 2010/2011


prof. Ing. Michael Šebek, DrSc.
vedoucí katedry




doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 12. 10. 2009

Declaration

I hereby declare that I have written my diploma thesis myself and used only the sources (literature, projects, SW etc.) listed in the enclosed references.

Prague, May 2010

A handwritten signature in blue ink, appearing to read "Benet", is written above a horizontal dotted line.

Signature

Acknowledgements

I would like to thank the chief of my diploma thesis Ing. Jiří Trdlička, for his patience, help and suggestions, which were very much appreciated.

I would like to special thank my family and my girlfriend for constant support and encouragement during the course of my studies.

Abstrakt

Tato práce se zabývá testováním a aplikací modulu ITEM. Jde o komunikační modul navržený pro použití v aplikacích pro bezdrátové senzorové sítě. Díky implementovaným protokolům TDMA (Time Division Multiple Access) a E-ASAP (Extended-Adaptive Slot Assignment Protocol) je modul schopný reagovat na dynamické změny struktury sítě. Původní verzi modulu je potřeba před použitím v reálných aplikacích důkladně otestovat a odstranit objevené nedostatky. Pro testování je nutné zajistit sběr dat ze sítě a jejich zpětné vyhodnocení. Nejprve je ověřena správná funkčnost modulu. K tomuto účelu byla vytvořena JAVA aplikace, která zachycuje a přehledně zobrazuje přesný sled událostí jednotlivých zařízení. Jejich vyhodnocením pak můžeme, vzhledem k použité struktuře sítě, rozhodnout o správnosti funkce. Po otestování správné funkčnosti jsou testovány vlastnosti sítě s použitím modulu ITEM. Mezi tyto vlastnosti patří využití kapacity komunikačního kanálu a doba potřebná pro doručení dat od zdroje k cíli. Pro srovnání výsledků byla zvolena původní verze modulu použitého pro sběr dat, který využívá CSMA protokol. Na závěr je vytvořena aplikace pro měření dat. Měření dat v různých místech sítě je nejčastějším požadavkem bezdrátových senzorových sítí.

Abstract

This thesis deals with testing and application of the ITEM module. It is a communication module designed for usage in applications for wireless sensor networks. Thanks to implemented TDMA (Time Division Multiple Access) and E-ASAP (Extended Adaptive Slot Assignment Protocol) protocols the module is able to react to dynamic network structure changes. The original version of the module must be thoroughly tested and discovered inadequacies must be removed before using in practical applications. Data must be collected from the network and then re-evaluated for testing. First, a correct function of the module is verified. For this reason a JAVA application was created. It captures and clearly presents exact sequence of events of individual network devices. With re-evaluation of these events we can determine correctness of the ITEM module function due to used network structure. After verification of correctness, performance of the network with the ITEM module is tested. This performance test contains channel capacity utilization and message delivery from source to target duration. For comparison of results, original version of module used for data collection, which uses CSMA protocol was chosen. Finally, application for data measurement was created. Data measurement in various places of the network is the most common requirement of wireless sensor networks.

Contents

1	Introduction	1
	1.1 Aim of this thesis	1
	1.2 Wireless sensor networks	1
	1.3 TinyOS and NesC	3
	1.4 Hardware	3
2	Theoretical considerations	6
	2.1 TDMA protocol	6
	2.2 CSMA protocol	7
	2.3 ASAP and E-ASAP protocols	8
	2.3.1 ASAP protocol	8
	2.3.2 E-ASAP protocol	10
	2.4 Collection protocol	11
3	Realization	13
	3.1 ITEM module description	13
	3.1.1 Original version	13
	3.1.2 Inadequacies of original version	15
	3.1.3 Current version	16
	3.2 Collection and ITEM module interconnection	21
	3.3 ITEM module testing	25
	3.3.1 Testing and visualization of function	25
	3.3.2 Testing of performance	31
	3.4 Data measurement application	36

4	Experiments and results.....	39
	4.1 Testing and visualization of function.....	39
	4.2 Testing of data measurement application.....	48
5	Conclusion	50
6	Future work	52
	References.....	53
	Contents of the DVD-ROM	55

List of Figures

Figure 1.1 Application of wireless sensor network (edited original from [4]).....	2
Figure 1.2 TelosB module and block diagram (taken from [5]).....	4
Figure 1.3 Tmote Sky module – top (left) and bottom side view (taken from [8]).....	5
Figure 2.1 TDMA example	6
Figure 2.2 Guard periods example	7
Figure 2.3 Frame length example	8
Figure 2.4 ASAP protocol example (taken from [13]).....	9
Figure 2.5 E-ASAP protocol example (taken from [13])	10
Figure 2.6 Collection module connection	12
Figure 3.1 ITEM module	14
Figure 3.2 ITEM module connection	15
Figure 3.3 Frame reduction example.....	16
Figure 3.4 ITEM data sending procedure	18
Figure 3.5 General data packet format	19
Figure 3.6 ITEM INF packet details.....	20
Figure 3.7 Collection and ITEM module interconnection.....	22
Figure 3.8 Collection module with ITEM data sending procedure	23
Figure 3.9 JAVA application, part for data reading	24
Figure 3.10 Interface for testing connection.....	26
Figure 3.11 ITEM visualization packet format	26
Figure 3.12 ITEM visualization packet details – type Inf	27
Figure 3.13 ITEM visualization packet details – type Slot start	27

Figure 3.14 ITEM visualization packet details – type Broadcast send	28
Figure 3.15 ITEM visualization packet details – type Broadcast receive	29
Figure 3.16 JAVA application for testing and visualization of ITEM function.....	30
Figure 3.17 Testing performance data packet (details sent between nodes)	32
Figure 3.18 Testing performance data generation procedure (in case with the ITEM module)33	
Figure 3.19 Testing performance data generation procedure (in case without the ITEM module).....	34
Figure 3.20 Testing performance data packet (details sent to computer).....	35
Figure 3.21 JAVA application for testing of ITEM performance	36
Figure 3.22 Data measurement packet details	37
Figure 3.23 JAVA application for data measurement	38
Figure 4.1 Test 1 network distribution	40
Figure 4.2 Test 1 network performance (network with the ITEM module)	41
Figure 4.3 Test 1 network performance (original Collection module).....	41
Figure 4.4 Test 2 network distribution	42
Figure 4.5 Test 2 network performance (network with the ITEM module)	43
Figure 4.6 Test 2 network performance (original Collection module).....	43
Figure 4.7 Test 3 network distribution	44
Figure 4.8 Test 3 network performance (network with the ITEM module)	45
Figure 4.9 Test 3 network performance (original Collection module).....	45
Figure 4.10 Test 4 network distribution	46
Figure 4.11 Test 4 network performance (network with the ITEM module)	47
Figure 4.12 Test 4 network performance (original Collection module).....	47
Figure 4.13 Test 1 application for data measurement	49

Figure 4.14 Test 2 application for data measurement	49
---	----

Chapter 1

Introduction

1.1 Aim of this thesis

Technologies of wireless sensor networks (WSNs) are being rapidly developed in these days. The possibility of arbitrary configuration has a great potential for usage in industrial, military and many others applications. Wireless sensor networks consist of spatially distributed autonomous devices called nodes. Individual nodes fulfill several functions (from data measurement to processing, forwarding and evaluating). Nodes are often placed in hostile or badly accessible places. For the purpose of ensuring long lifetime of individual nodes, we must minimize their energy consumption. The highest consumption is caused by data transmission in the network.

At the Czech Technical University in Prague in the Department of Control Engineering was created a software module called ITEM (Integrated TDMA E-ASAP Module). This module controls forwarding of the data in the network in a way that there are no collisions, which are common problem at this type of networks.

Our first target of this thesis is identification of inadequacies of the ITEM module and their elimination. We also have to carry out such corrections that enable usage of the ITEM module in practical applications. We must thoroughly test the module before we can use it in practical applications. Another objective of this thesis is to create visualization of function of tested module. The last aim of this work is implementation of application for data measurement.

1.2 Wireless sensor networks

A great expansion and early successes of wireless sensor networks show that it has the opportunity to become a very useful tool, such as the Internet. Just as the Internet allows us faster, easier access to data and information from the digital world, sensor networks expand our ability to access data from the physical world [1].

The basic element of the network is a device called node. The node consists of a processing unit with limited computational power and limited memory, sensors, a communication device (usually radio transceiver) and a power source. These nodes are commonly located in certain areas to monitor a specific phenomenon. These areas are often remote or hostile. A power source is typically formed of batteries, but unlike cell phones or wireless laptops periodic recharging is not possible. For this reason, great emphasis is on low power consumption. Most power is consumed by data transmission and thus sensor networks often form wireless ad-hoc networks [2], where each node supports multi-hop routing algorithm (data forwarding to the base station through other devices in the network).

Special requirements of sensor network applications and resource constraints in sensor network hardware platform determine the properties of used operating systems. They are typically less complex than general-purpose operating systems. Probably, the most popular operating system for wireless sensor networks is TinyOS. Programs for TinyOS are written in special programming language called NesC, which is an extension of the C programming language.

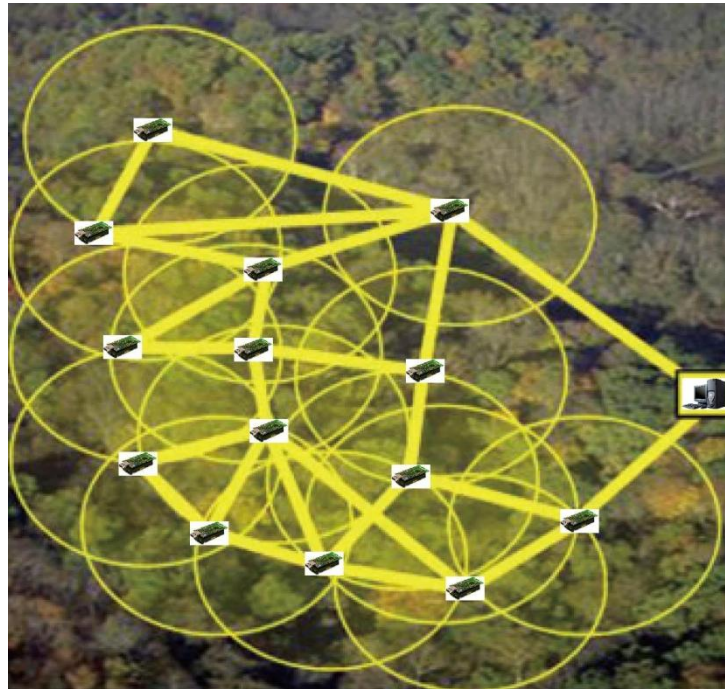


Figure 1.1 Application of wireless sensor network (edited original from [4])

1.3 TinyOS and NesC

TinyOS [3] is an operating system designed especially for usage in wireless sensor networks. Its programming model is customized for event-driven applications. TinyOS provides a set of reusable components. These components provide and use interfaces for their wiring. These interfaces are the only point of access to the component.

Two sources of concurrency are tasks and events. Task is deferred computation mechanism which runs to completion. Tasks do not preempt each other. A component can use task when timing requirements are not strict. Events also run to completion, but it can preempt the execution of a task or another event.

Only one application runs on node at a time. Deep tying of application with used hardware leads to three important requirements. First requirement is that all resources are statically known. Second, applications are built of a set of reusable system components, connected with application specific code. Third, a hardware/software boundary varies depending on the application and hardware platform.

Used programming language NesC simplifies the application development, reduces the code size and eliminates many sources of potential bugs by performing a whole-program optimization and compile time data race detection. NesC distinguishes between two types of components: modules and configurations. The modules provide application code and they implement one or more interfaces. The configurations wire used components together.

The NesC programming language meets unique requirements for programming of wireless sensor network applications. Success of the component based model is shown by the way in which components are used in the TinyOS code.

1.4 Hardware

Designs of devices for wireless sensor networks vary from application to application. We meet with nodes size as a coin to nodes size as a personal computer, with cheap nodes for simple applications to expensive, sophisticated devices for demanding applications. We use TelosB and Tmote Sky modules in this work. Both modules are almost identical. Tmote Sky module will be described in detail.



Figure 1.2 TelosB module and block diagram (taken from [5])

The main parts of the module are a microcontroller, a radio transceiver, a power source, an external memory and one or more sensors (we did not have sensors available in this work). The brain of the node is ultra low power Texas Instruments MSP430 F1611 microcontroller featuring 10kB of RAM and 48kB of flash. This 16-bit RISC processor has an internal digitally controlled oscillator that operates up to 8Mhz. The features of MSP430 F1611 are presented in detail in the Texas Instruments MSP430 x1xx Family User's Guide [6]. The Chipcon CC2420 (IEEE 802.15.4 compliant) radio takes care of the wireless communication. The CC2420 has programmable output power and provides a digital received signal strength indicator (RSSI). Usage of CC2420 is available in Chipcon's datasheet [7]. The node contains USB port for serial communication. The power source consists of two AA batteries. The serial code flash ST M25P80 40Mhz is used for data storage. The flash shares SPI communication lines with the CC2420 transceiver. Tmote Sky module contains two expansion connectors for additional devices (analog sensors, LCD displays and digital peripherals) and connection for two photodiodes and for a humidity/temperature sensor.

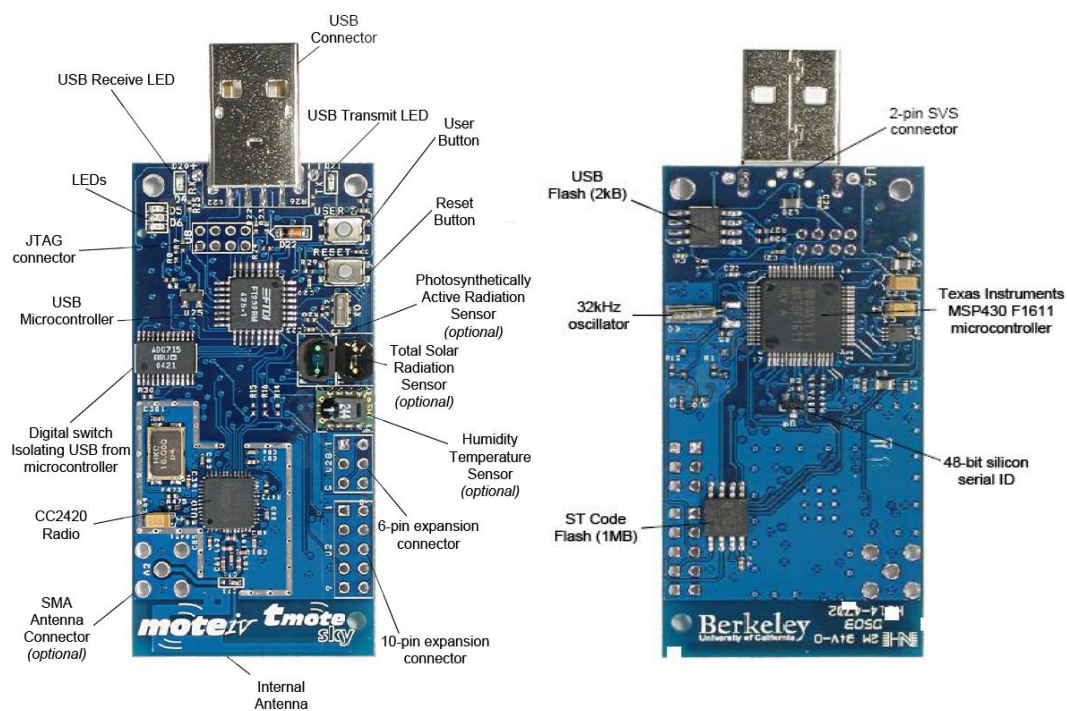


Figure 1.3 Tmote Sky module – top (left) and bottom side view (taken from [8])

Chapter 2

Theoretical considerations

2.1 TDMA protocol

Time Division Multiple Access is a shared medium access method [9]. This method allows several users to share one medium by dividing time into time slots.

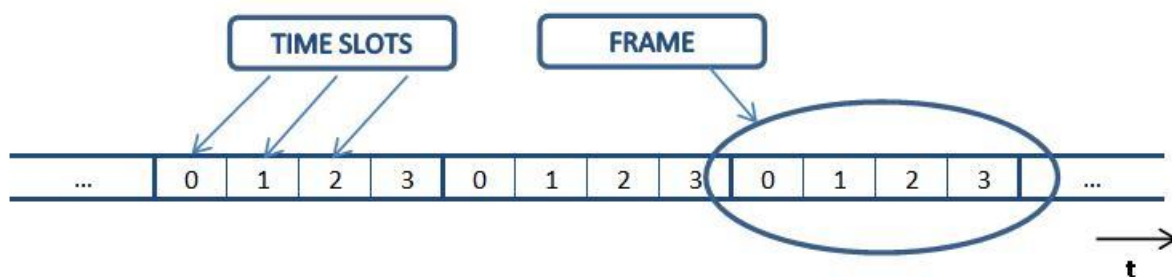


Figure 2.1 TDMA example

These successive repeating time slots are assigned to those devices, which access the medium. Each time slot is owned only by one device. Each device can access shared medium only in own time slots. Fulfillment of these assumptions guarantees prevention of multiple accesses at the same time and occurrence of collisions. A group of several time slots is called a frame.

This medium access method requires accurate time synchronization of all devices. So-called guard periods are used for inaccuracies in time synchronization. Medium access is permitted in these periods. This restriction prevents time slot overlap and possibility of collision occurrence.

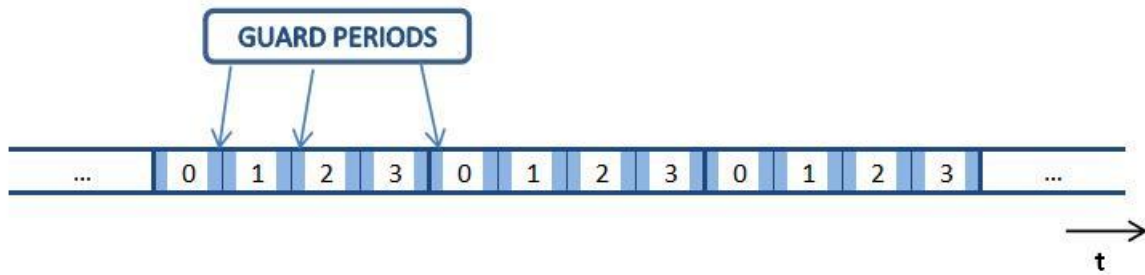


Figure 2.2 Guard periods example

There are many options of time slot assignment. Time slot assignment used in this thesis is inspired by E-ASAP protocol, which will be described later in this text.

2.2 CSMA protocol

Carrier Sense Multiple Access is another shared medium access method. This method is based on probabilistic medium access protocol. Each device verifies medium availability before a data sending can start. Data can be sent only when medium is free. Using CSMA method doesn't prevent from collision occurrence (simultaneous medium access). There exist CSMA modifications which try to minimize collision occurrence or even completely remove collisions [10].

One of these modifications is CSMA/CA (CSMA with Collision Avoidance). If a device, which wants to send a data, detects occupation of a channel, it waits random time before data sending. This random time reduce a probability of a collision. This modification of CSMA is often used in wireless sensor networks [11].

Another possible modification is CSMA/CD (CSMA with Collision Detection). Sending devices are able to detect collision and stop sending immediately. Another attempt to send data runs through random time delay. Stopping immediately after collision detection results in more efficient capacity utilization, because there is no wasting with bandwidth usage for sending entire collision packet. CSMA modification with collision detection is not used in wireless networks [12].

Furthermore, there exists modification CSMA/BA (CSMA with bitwise arbitration) known also as CSMA/CR (CSMA with collision reduction). All devices are assigned

a priority. When there is collision detection, the device with higher priority is preferred for data sending.

2.3 ASAP and E-ASAP protocols

Time slot assignment between network devices can be scheduled in many ways. E-ASAP protocol (Extended Adaptive Slot Assignment Protocol) [13] deals with this issue. This protocol extends ASAP protocol (Adaptive Slot Assignment Protocol). E-ASAP in comparison with ASAP improves channel utilization. Principles of both protocols will be described in this chapter.

2.3.1 ASAP protocol

ASAP protocol sets frame length of new node (a node that is currently connected to the network) according to frame length of nodes in its contention area (set of nodes, which can cause sent packets collision) and minimizes number of unassigned slots. This mechanism improves channel utilization. TDMA format in ASAP is shown in Figure 2.3.

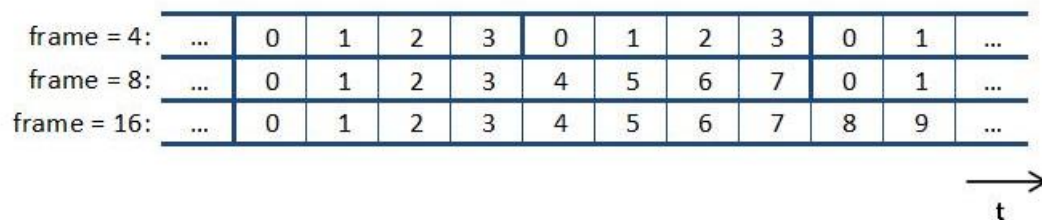


Figure 2.3 Frame length example

Frame length of each node can be changed dynamically (doubled or halved) according to slot assignment in its contention area. This prevents collisions of packets sent by devices with different frame length. First slot is reserved for sending control packets of newly connected devices so there are no data packet sent.

Each node keeps information about own frame length and assigned own slots, as well as information about assigned slots of neighbor nodes and information about assigned slots of hidden nodes (nodes at distance of two hops).

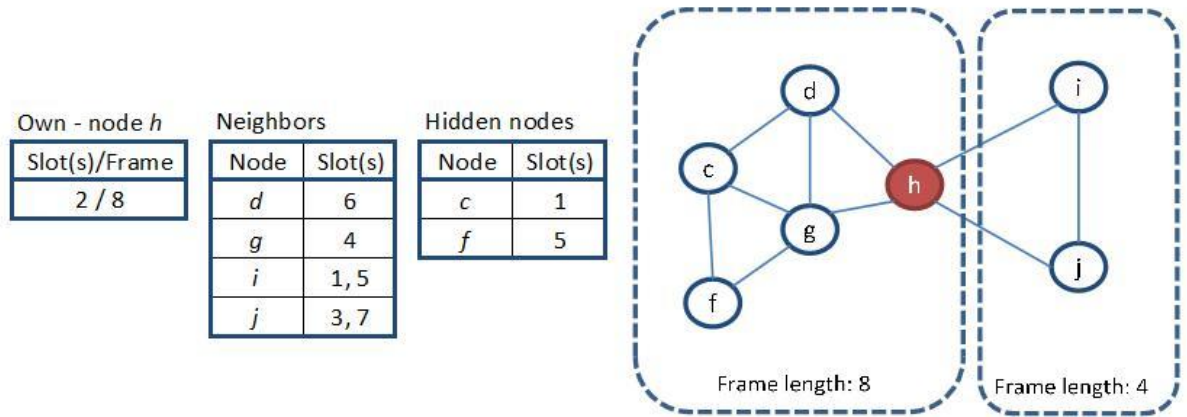


Figure 2.4 ASAP protocol example (taken from [13])

Details about neighbor and hidden nodes are extracted from collected information packets that each device sends in own assigned time slot. Each node sends two types of packets. The first type is the above-mentioned information packet that contains details about assigned slots of sender and his neighbors. The second type is data packet that contains details about sender's frame length, sender's current slot, maximal frame length among the sender and its neighbors and data.

Slot assignment procedure for newly connected devices is as follows. First, a new node collects information sent by neighbor nodes. This node then sets the frame length as the maximum frame among all nodes in its contention area. If there are unassigned slots, they can be assigned to the new node. If all slots are occupied, but there are nodes that own more slots, slot is released from the node which owns the largest number of slots. New node then assigns newly created, free slot itself. If there are no free slots or any device owns more slots, the new node doubles the frame length and assigns free slot itself. The first slot in frame is reserved, so doubling the frame length and copying slots from lower to the upper half of the frame creates a free slot on the position of the first slot in upper half of the frame.

In the slot assignment conflicts can occur (e.g. if we connect new node between two nodes with the same own slot, out of radio range). Conflict solution follows three methods. First way is to delete the conflicting slot. When nodes with conflicting slots own also correctly assigned slots, conflicting slots are simply deleted. Second way is to divide conflicting slots

between nodes. When each node owns only conflicting slots and there is no possibility to solve the conflict in current frame length, frame length is doubled and slots are divided between nodes.

2.3.2 E-ASAP protocol

Although ASAP improves channel utilization compared with conventional slot assignment protocols, the frame length in whole network tends to grow (because the initial set of maximum frame length). Each node assigns own slot at minimum frame length in which there are no conflicts in E-ASAP protocol. This method improves channel utilization in comparison with ASAP.

First slot is reversed similarly as in ASAP protocol. E-ASAP protocol differs in retained information. Information about assigned slots of neighbor and hidden nodes extends information about frame length of these nodes.

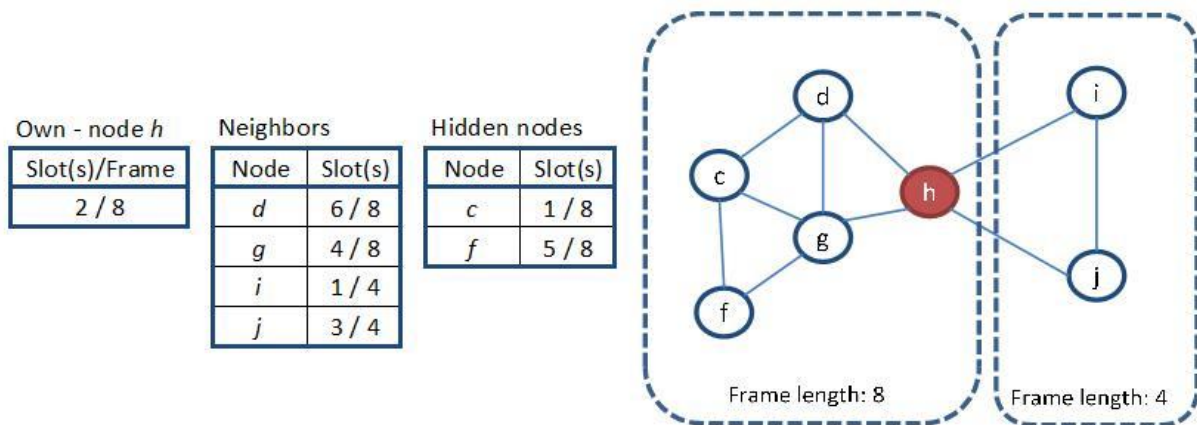


Figure 2.5 E-ASAP protocol example (taken from [13])

Slot assignment procedure for newly connected devices is similar to ASAP protocol. New node collects information sent by neighbor nodes, then sets the minimum frame length (four slots) and at the end assigns a free slot itself like in ASAP.

Detection and solution of conflicts is the same as in ASAP.

2.4 Collection protocol

Typical application for wireless sensor networks considers free placement of nodes in free space. There is only a small amount of nodes that are connected to the base station. It is usually required to evaluate data from all nodes in the network. Data must be transported from source to base station. For this purpose various mechanisms are proposed for data collection (e.g. [14], [15] and [16]). One of these mechanisms is Collection protocol [17] included in applications for TinyOS 2.x.

The principle of Collection protocol is base on building of one or more collection trees. Each node in the network is a part of only one collection tree. Nodes can send own generated data or resend received data. Collection protocol tries to deliver data from source to at least one root in the network. There are no guarantees of data delivery.

A node can perform four different roles. Nodes that generate data for sending are called producers. Nodes that overhear messages are called snoopers. Forwarding message can be modified. Nodes that modify forwarding messages are called in-network processors. Nodes that only receive data from network have the last role. These nodes are called consumers.

Implementation of Collection protocol consists of three main components: Link estimator, Routing engine and Forwarding engine. Link estimator estimates link quality among neighbors. Link quality is represented by a number, whose value should be in range [0, 65535]. Smaller values signal better link quality. Routing engine calculates a route from the source to the root of a collection tree. Roots can be changed dynamically. Routing engine keeps information about neighbor nodes that can build a route to the root of the tree. The best candidate is chosen according to link quality detected in Link estimator. Forwarding engine attends to message transmissions. It evaluates retransmissions, duplicate suppressions, packet timing and loop detection. CSMA protocol, which is described in chapter 2.2, is used for access to the communication channel.

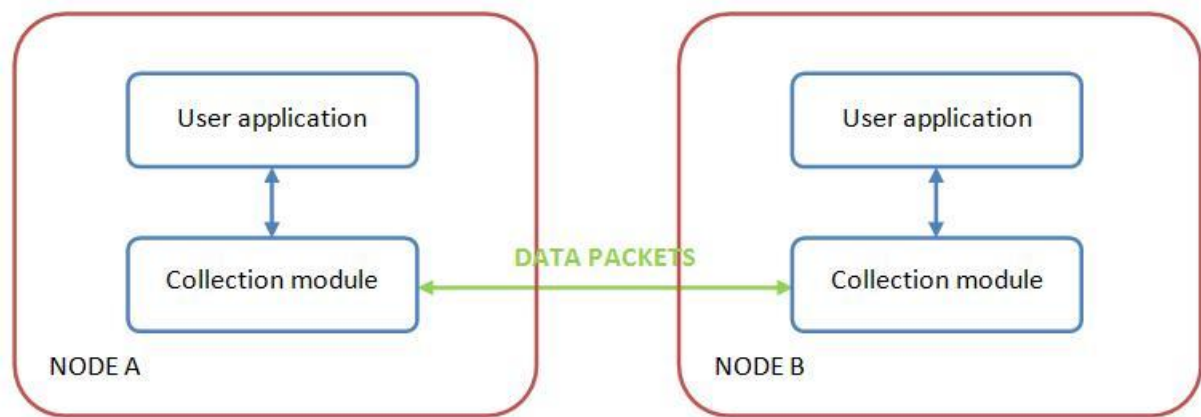


Figure 2.6 Collection module connection

Chapter 3

Realization

3.1 ITEM module description

3.1.1 Original version

The first version of ITEM module [18] was created at the Czech Technical University in Prague in the Department of Control Engineering in order to build a communication module for wireless sensor networks. This module should provide collision free (elimination of data packets interference) data transmission in the network. The main part of the communication module was designed according to E-ASAP protocol that uses the principle of TDMA protocol. First version was designed for TinyOS 1.x operating system. TinyOS 1.x has been replaced by a new version TinyOS 2.x. We have rewritten first version of ITEM for TinyOS 2.x operating system [19]. TinyOS 2.x is used in wireless sensor networks until today.

Structure and properties of communication module ITEM will be described in this chapter. The whole system is shown in Figure 3.1. It consists of several smaller, independent parts. Each part of the module performs the corresponding function. With this layout the module can be easily developed, tuned and tested.

The first part of the module is called TimeSync. We need accurate time synchronization of all devices for correct time slots sequence in TDMA protocol. TimeSync module fulfills these requirements. Time synchronization is based on simple averaging algorithm. Each device sets its own local time at connection to the network according to neighbor nodes. Local time is then corrected according to:

$$t_{local} = \frac{t_{remote} + t_{local}}{2}.$$

SimpleTime module was created for local time manipulations.

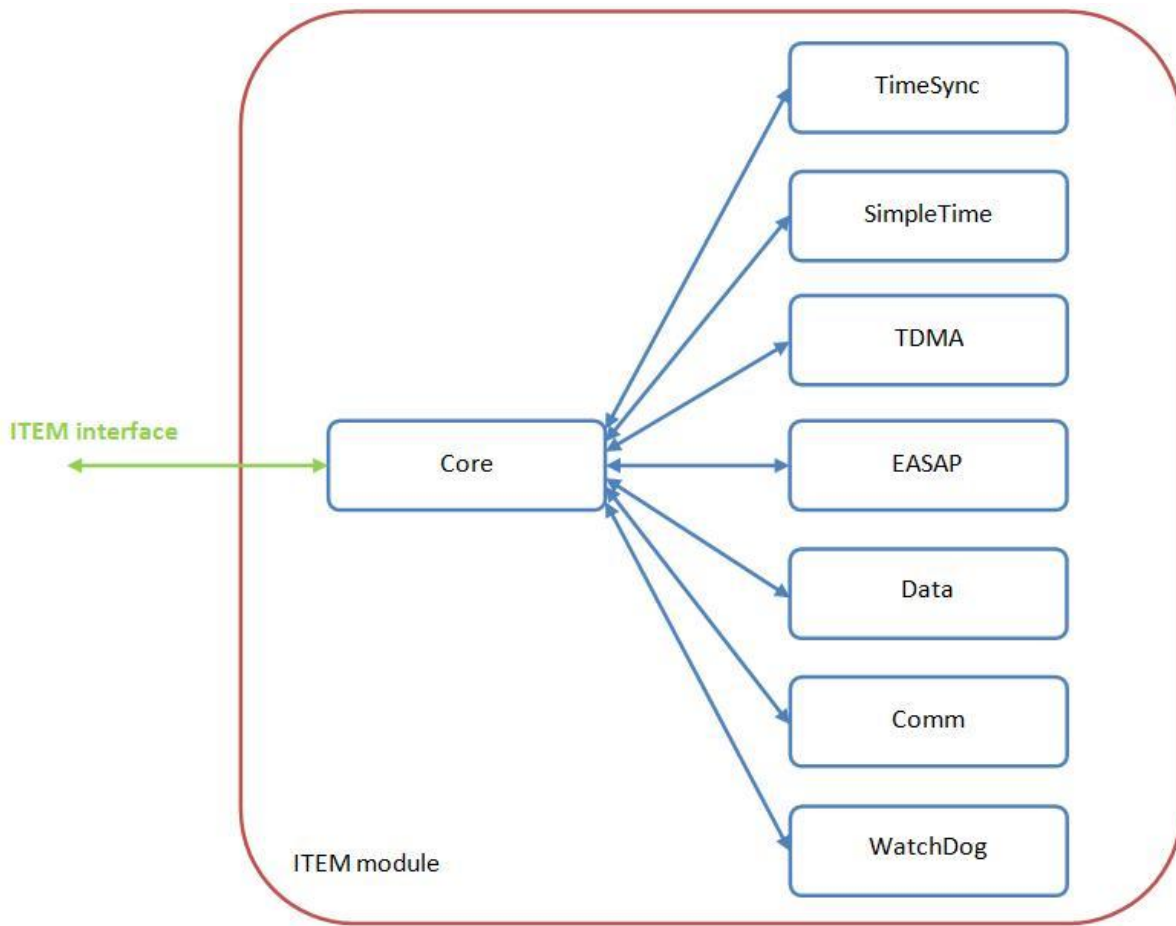


Figure 3.1 ITEM module

If the device has correct local time, the principle of TDMA protocol can be applied. Part called TDMA performs time division into time slots. Information about local time and frame length must be known to generate correct time slot number. When time is divided into time slots, time slots must be correctly assigned to devices. Wrong distribution of time slots can cause collision in data transmission. Module called EASAP according to implemented E-ASAP protocol provides correct distribution. E-ASAP protocol is resistant to dynamic changes in the network.

Messages are sent and received by a module called Comm. This module provides radio and serial communication. In this version, the ITEM module distinguishes between two message types: Information packets (INF) and Data packets (DAT).

Data can be sent only in node's own time slots. That means that data generated out of these time slots must be stored. A module called Data was created for this purpose. The Data module consists of two data queues. One queue keeps data for sending, second queue keeps received data until a user processes them.

ITEM module contains module called WatchDog. WatchDog monitors whole system functionality. If this module detects an irregular running of a program, it restarts the device and sets the program into the initial state.

The last part of ITEM module is called Core. Core connects all these modules into one functional module. It provides an interface for superior application. Users do not need worry about data sending. They only choose a number of owned slots and then insert sending data and process received data. The ITEM module ensures correct data delivery.

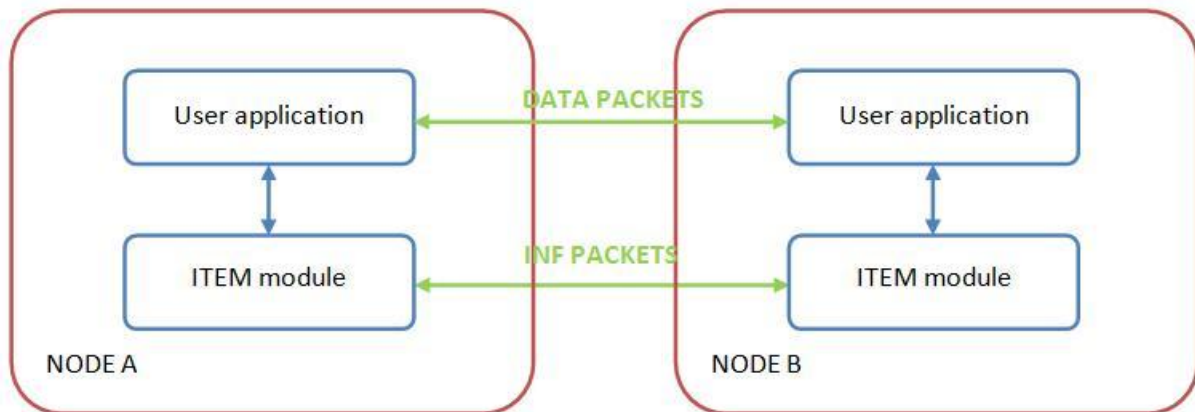


Figure 3.2 ITEM module connection

3.1.2 Inadequacies of original version

Original version contains some inadequacies. One of these inadequacies is occupation of the same slot by nodes connected to the network in one place at the same time. Connected nodes receive the same information packets, but they don't know anything about one another. This leads to occupation of the same slot by connected nodes and occupation of the same slot leads to collisions.

One of inadequacies is possibility of sending only one message per time slot. Most of the time inside the slot is unused. This limitation results in poor channel utilization.

Improved capacity utilization of the communication channel can be achieved by better solution of frame length reduction. In original version, the frame length reduction is possible only if there are no assigned slots in upper half of frame. We can see a small example of this case in Figure 3.3. There are four nodes in the network. Assigned slots are: 1, 2, 5 and 6. Frame length is 8. If we remove node with own slot number 2 from the network, there is no frame length reduction because it is not possible. If nodes with own slots number 5 and 6 replace these slots with free slots in lower half of frame, reduction of frame length is possible. The reduction of frame length results in better channel utilization.

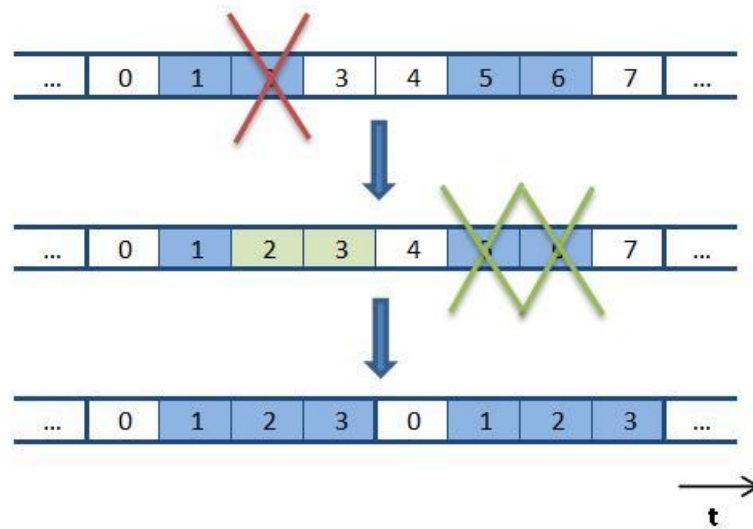


Figure 3.3 Frame reduction example

All messages sent by ITEM module are broadcasted. A superior application has no possibility of sending messages only to given device in range.

Last recognized inadequacy is sending messages over one, ITEM's defined, radio interface. Parts of data packet (header and footer) are created according to used interface. This mechanism of data sending complicates connection of ITEM module with superior applications that divide and process packets according to these parts.

3.1.3 Current version

The current version is substantially modified in a comparison with the original. Motivation for these modifications was to enhance performance and eliminate the inadequacies of the original version. The main idea of the ITEM module was maintained.

Adjustments relate primarily to various parts of the module. Important change addresses a way to send messages. Sending messages with the ITEM module is found to be unsuitable for practical applications (e.g. using application for data collection with ITEM). This way of sending data satisfies using of applications that use only one type of messages and only one radio interface. If application uses more types of messages and more radio interfaces, then it is easier for data sending to use radio interfaces used in application. In current version the ITEM module doesn't send data packets. Data sending must be implemented in superior application (where necessary radio interfaces can be used). The ITEM module only signals times for data sending. Procedure for data sending is presented in Figure 3.4.

The diagram in Figure 3.4 doesn't include idle times at the beginning of time slots. These idle times delay sending data due to possible inaccuracies in time synchronization. Information packets are not sent in each own slot as shown in figure. Information packets are sent after defined number of own slots. The ITEM module computes remaining time in own slot. It compares computed remaining time with time used for sending the longest message and evaluates the possibility of sending another message. This leads to an improvement of the original version. The ITEM module can arrange sending more messages during own slot in this version. Data sending is moved to the superior application that uses necessary radio interface. This results in possibility of messages addressing (removes restriction for broadcasted messages). This way of sending data also reduces data types used in ITEM module. There are no data packets (DAT) in ITEM. Data queues in module called Data are not needed, but this module was not removed. Data module still exists, but there is only one data queue. This queue is designed for storage data from superior application. Function for data storage and information about data queue properties were built into ITEM interface.

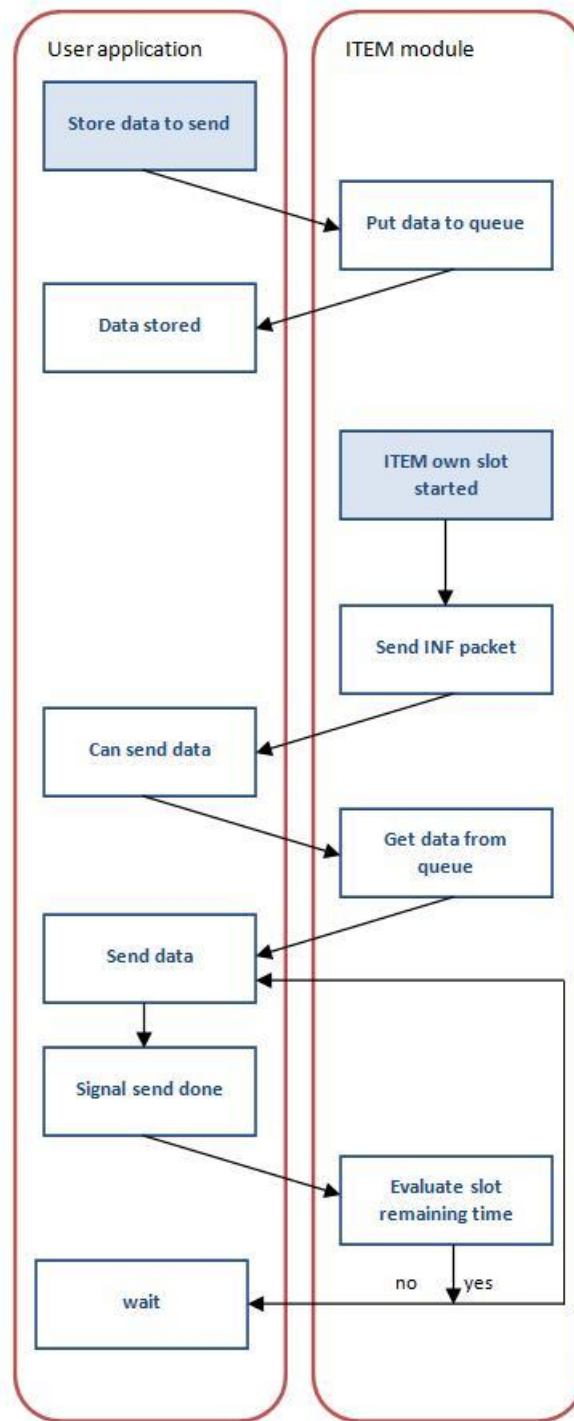


Figure 3.4 ITEM data sending procedure

Other changes affected computation of the local time. 64-bit local time, composed of two 32-bit numbers, was replaced by 32-bit local time. This modification simplifies local time computations, saves memory and reduces the length of the information packet.

Original version doesn't deal with bad time slot assignment. Each device now evaluates bad slot assignment according to details in received information packets. If a device detects that there is another device with the same slots in contention area, it analyses this situation and release collision slot. A new time slot is assigned with random delay. There is a Random component in TinyOS 2.x for random delay generation. This way of collision solution also solves restrictions of sequent node connection. When assignment of bad slot is detected, slot is released and then a new slot is assigned with random delay. Random delay simulates sequent connection of nodes.

Current version of ITEM solves correct frame length reduction. If the following relationship is satisfied, assigned slots from upper half of frame are released, frame length is reduced and nodes that released upper slots assign new free time slots themselves.

$$\frac{\text{number of assigned slots in frame}}{\text{frame length}} \leq \frac{\text{frame length}}{2} - 1.$$

This reduction results in better channel utilization. A function for determination of the quantity of owned slots by node was added to ITEM interface.

An actual reduction of frame length is evaluated according to:

$$\frac{\text{number of assigned slots in frame}}{\text{frame length}} + \frac{\text{number of reserved slots in lower half of frame}}{\text{frame length}} \leq \frac{\text{frame length}}{2} - 1.$$

The new version implements a possibility of time slots reservation. Time slots marked as reserved can't be assigned to any node. These slots serve for different purposes (e.g. sending test messages). Information about reserved slots was included in the information packet as shown in Figure 3.6. Each data packet sent to the serial link of the computer is in format presented in Figure 3.5.

00	DEST ADDR	SOURCE ADDR	MSG LENGTH	GROUP ID	HANDLER TYPE	DATA
1 B	2 B	2 B	1 B	1 B	1 B	n B

Figure 3.5 General data packet format

Output meaning:

00 – Leading byte,
 DEST ADDR – Destination address,
 SOURCE ADDR – Source address,
 MSG LENGTH – Data length,
 GROUP ID – Message type,
 HANDLER TYPE – Message type,
 DATA – Data itself.

DATA TYPE	DATA LENGTH	TIME STAMP	RESERV SLOTS	ID	FRAME	NR OF SLOTS	SLOTS	NR OF NODES	NODES INFO
1 B	1 B	2 B	8 B	2 B	1 B	1 B	n B	1 B	m B

Figure 3.6 ITEM INF packet details

Output meaning:

DATA TYPE – Data type,
 DATA LENGTH – Length of information packet,
 TIME STAMP – Time of packet generation,
 RESERV SLOTS – Reserved slots,
 ID – Node Id,
 FRAME – Node actual frame length,
 NR OF SLOTS – Quantity of assigned time slots,
 SLOTS – Assigned time slots,
 NR OF NODES – Quantity of neighbors,
 NODES INFO – Neighbors info (Id, frame length, nr. of slots, slots).

3.2 Collection and ITEM module interconnection

As mentioned in chapter 2.4, Collection module for collecting data from the network is included in applications for operating system TinyOS 2.x. Using this module ensures data delivery from data source to the base station. Sending and forwarding messages is based on the CSMA protocol [10]. If we want to use Collection module together with designed ITEM module, Collection module must be edited. These modifications are described in this part of the thesis.

The simplest solution is creation of a new module that interconnects these two modules. Newly created module is called CollToItem. Interconnection is shown in Figure 3.7.

Prepared data for sending to the root of the Collection tree are generated in User application and then inserted into the Collection module. Collection module passes received data to User application when the node is marked as the root of the tree. Messages that we don't want to send to the root of the tree must be inserted directly into the CollToItem module.

Original version of the Collection module sends and receives data packets as shown in Figure 2.6. But we want to send data by ITEM module, which ensures that the data will be sent at the right time (in correct time slot). Direct data sending from Collection module is replaced by passing data to the CollToItem module. Receiving data is conserved in Collection module. Message acknowledgement is for this kind of communication (using TDMA) turned off. It is because of the possibility of sending more data in own slot. A transmitting device does not have to wait for confirmation from receiving device, which can reply only in own slot with the ITEM module.

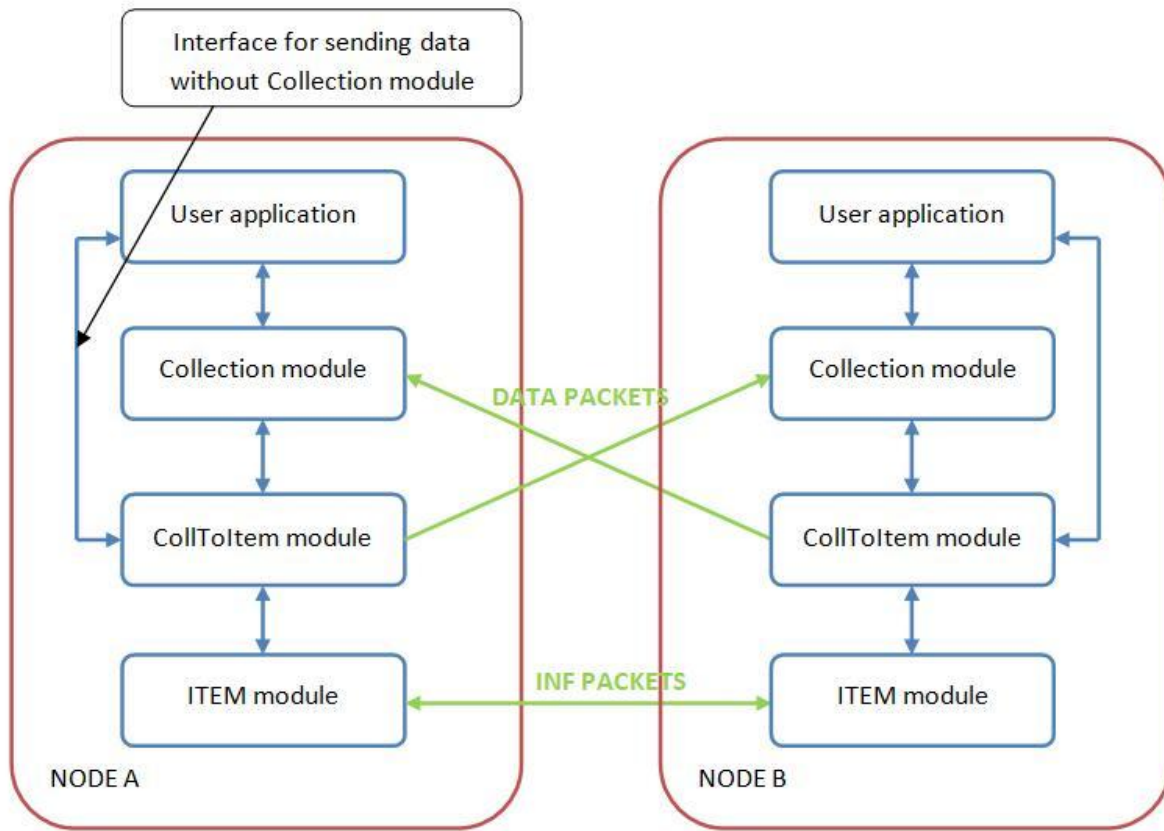


Figure 3.7 Collection and ITEM module interconnection

The CollToItem module receives messages for sending from Collection module and from User application. Received messages are marked with corresponding type (estimation or forward message from Collection module, broadcast message from User application) and stored in data queue provided by the ITEM module. If data queue is full the CollToItem module indicates unsuccessful receipt of data. ITEM module signals to the CollToItem module right time for sending. Data packet is removed from data queue and sent over the same radio interface which is used in Collection module or User application for receiving.

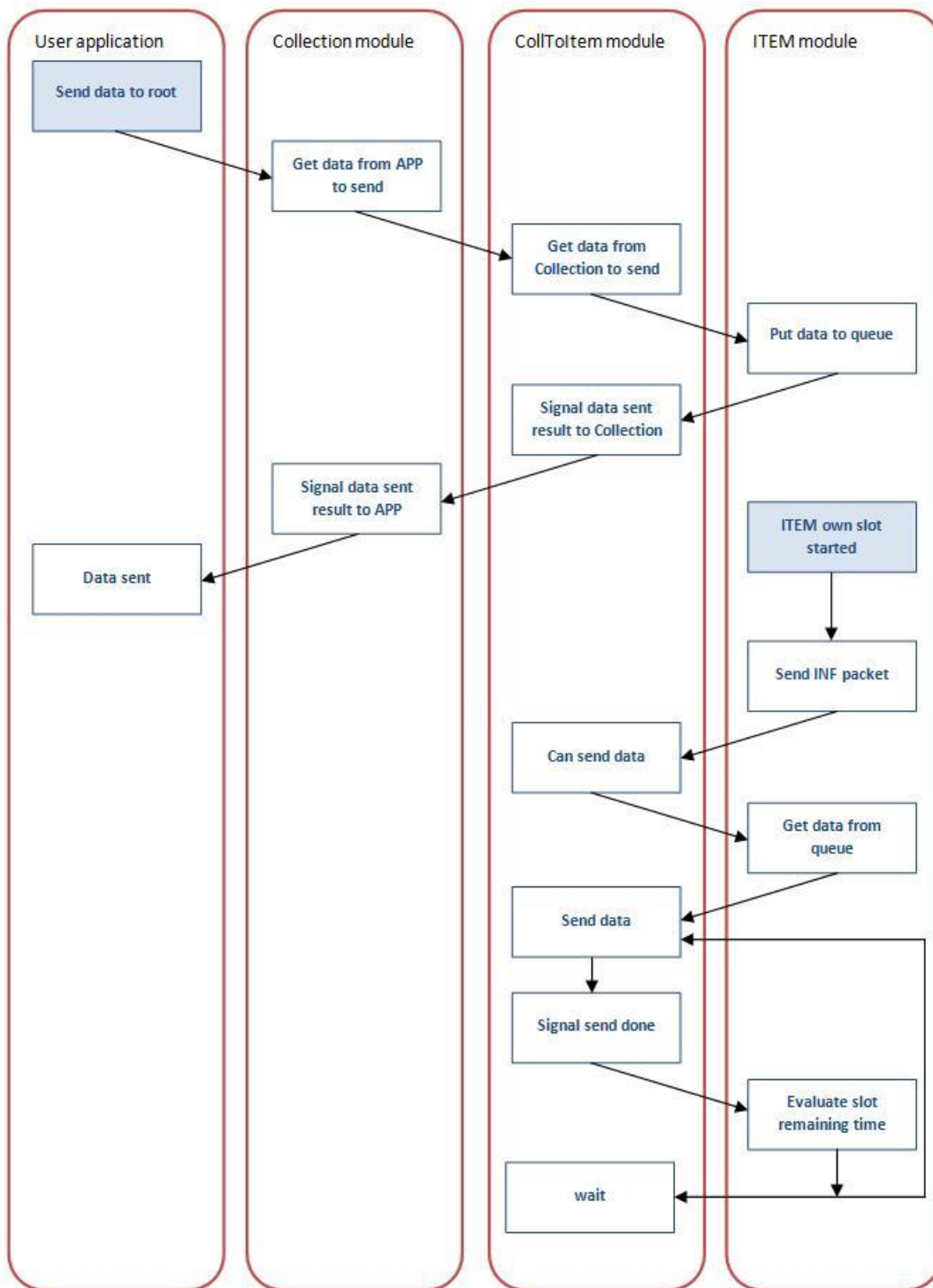


Figure 3.8 Collection module with ITEM data sending procedure

Data generated at various points in the network are sent to the root of the tree. The root forwards received messages to a serial link of the connected computer. A JAVA application

for processing and storing data in computer was created. The application has a simple graphical user interface. It is divided into two parts. Upper part for testing the ITEM module and displaying measured data will be described later. Lower part is responsible for retrieving data from the serial link of a computer.

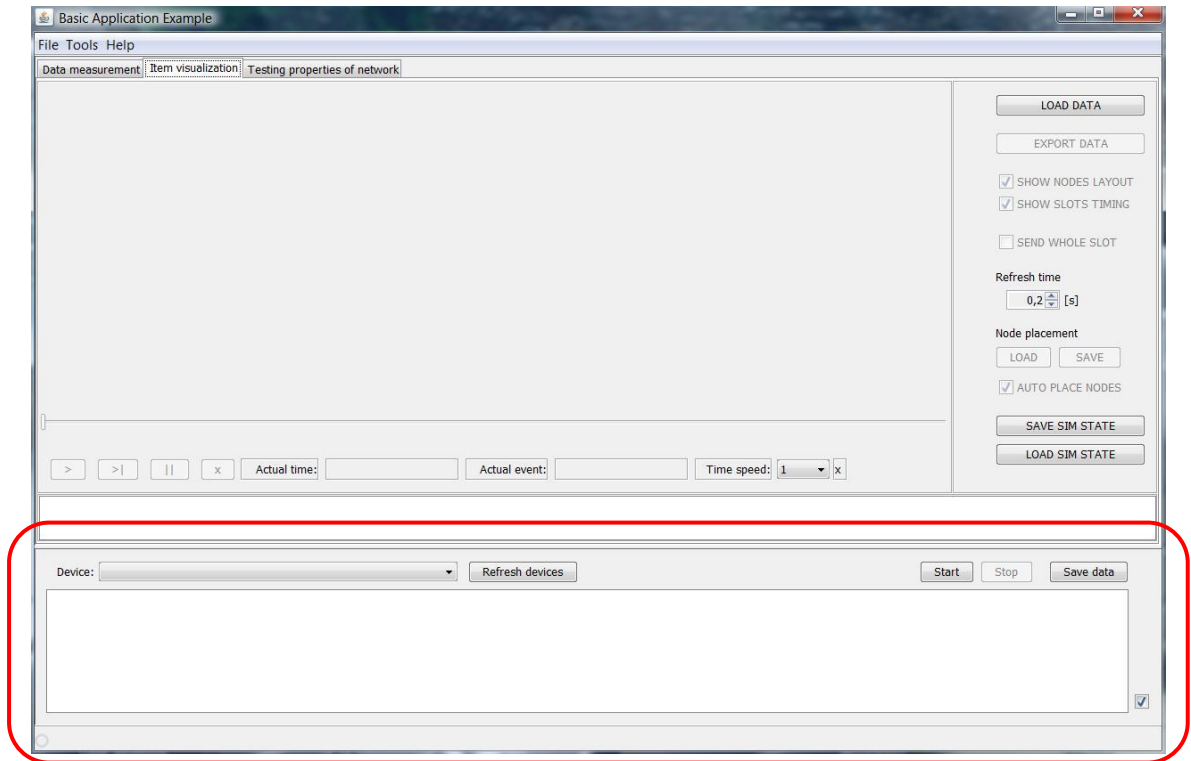


Figure 3.9 JAVA application, part for data reading

Connected devices are automatically loaded in the combo box Device after running the application. We need to update combo box with Refresh devices button after connection or disconnection a device at application runtime. JAVA application used for reading data packets from serial is provided in sources in the TinyOS 2.x with name Listen. Listen application is launched and its output is redirected to our application when connected device is selected and data reading is started. Received data packets are displayed in the form:

Date,Time: Received data packet

Check box in lower right corner is used for turning on or off automatic scrolling to the last received data packet. Received data packets can be saved in file.

3.3 ITEM module testing

Modified ITEM module must be thoroughly tested. To verify behavior of the ITEM module we need to know information about states of all participating devices in the network. Data collection is described in previous chapters. Principle of testing methods and evaluation of results is described in this part. Tests can be divided into two groups. Functionality of the ITEM module was in the first group of testing. Second group consists of tests of performance of the ITEM module. The achieved results are described in chapter 4.

3.3.1 Testing and visualization of function

It is very hard to test the ITEM module as a whole. We need to evaluate a lot of information on each device in the network to confirm the correct functionality. This information includes:

- Slot timing
- Slot assignment
- Frame length setting
- Sent and received messages times

We need to get this information into User application. User application then sends the data to the root of the tree through Collection module. There are two interfaces for getting information into User application. First interface, called ItemTest, obtains information from the ITEM module. Second interface, called CollToItemTest provides information about sending and receiving messages from the CollToItem module. Interface and module interconnection is shown in Figure 3.10.

We can made visualization of the ITEM module function from the measured data. We now describe a way of generating the necessary information. Data packet format is the same as the one presented in Figure 3.5. Item DATA in data packet is different and its format is shown in Figure 3.11.

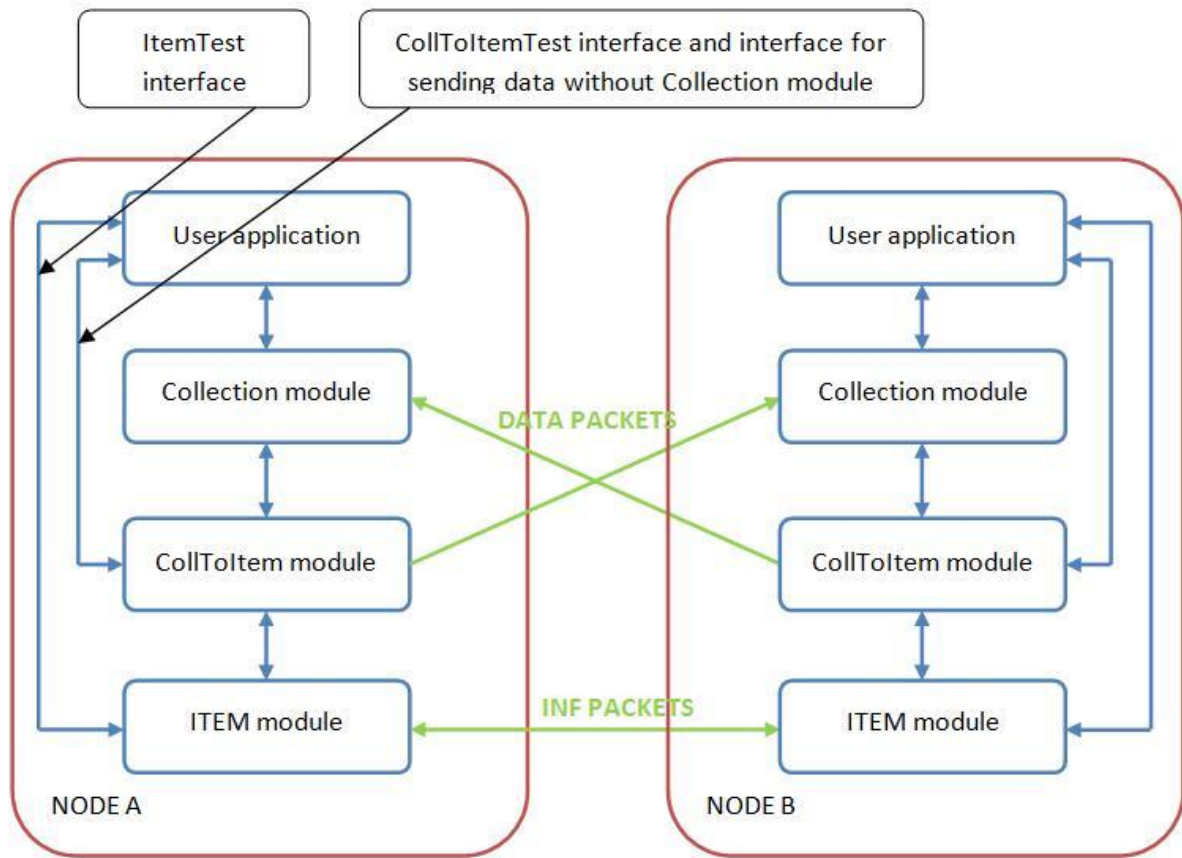


Figure 3.10 Interface for testing connection



Figure 3.11 ITEM visualization packet format

Output meaning:

APP TYPE – Application id (02 = ITEM visualization),

NODE ID – Node Id,

TIME – Time of packet generation,

INFORMATION – Generated information.

Generated information is divided into four types. Types correspond to the properties that we need to know for visualization. First type is called Inf packet. Its structure is as shown in Figure 3.12.

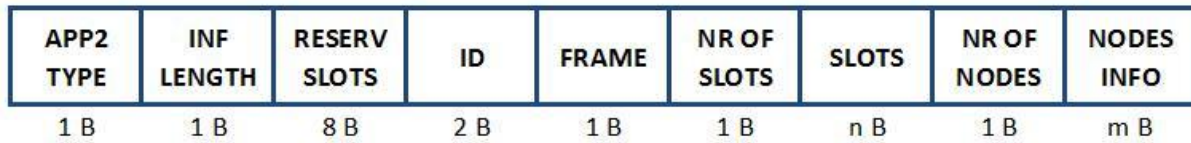


Figure 3.12 ITEM visualization packet details – type Inf

Output meaning:

APP2 TYPE – Type of information (01 = Inf packet),

INF LENGTH – Inf packet length,

Rest of the data packet is a copy of real INF (copy starts from reserved slots item) packet used in the ITEM module (Figure 3.6).

This type of information contains all items preserved in the EASAP module. It is an exact copy of a real INF packet that the ITEM module sends. This information is generated in time when a real INF packet is sent. Second type is called Slot start packet. Its structure is as shown in Figure 3.13.

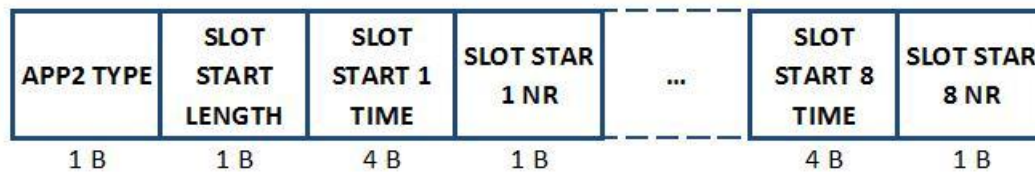


Figure 3.13 ITEM visualization packet details – type Slot start

Output meaning:

APP2 TYPE – Type of information (02 = Slot start packet),

SLOT START LENGTH – Slot start packet length,

SLOT START 1 TIME – Time of slot start 1,

SLOT START 1 NR – Number of slot 1,

⋮

SLOT START 8 TIME – Time of slot start 8,

SLOT START 8 NR – Number of slot 8.

This type carries information about times of slot starts. Eight consecutive slot starts are contained in one data packet for reducing the quantity of generated data packets. Third type of information is called Broadcast send packet. Its structure is presented in Figure 3.14.

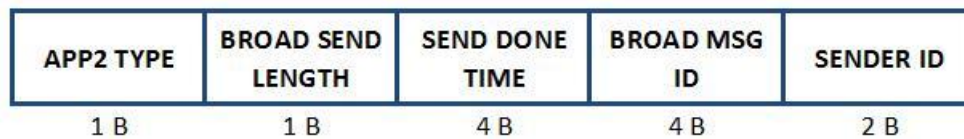


Figure 3.14 ITEM visualization packet details – type Broadcast send

Output meaning:

APP2 TYPE – Type of information (03 = Broadcast send packet),

BROAD SEND LENGTH – Broadcast packet length,

SEND DONE TIME – Time of radio send done signal,

BROAD MSG ID – Id of broadcasted message

SENDER ID – Sender Id.

Each node in the network generates broadcast messages. Information about time and duration of broadcasts is stored. Last type of information is called Broadcast receive packet and it holds information about times of received broadcasts. One data packet contains information about four broadcasted messages. Its structure is presented in Figure 3.15.

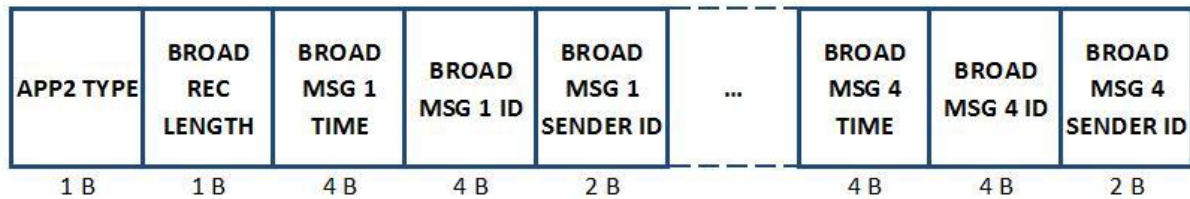


Figure 3.15 ITEM visualization packet details – type Broadcast receive

Output meaning:

APP2 TYPE – Type of information (04 = Broadcast receive packet),

BROAD REC LENGTH – Broadcast packet length,

BROAD MSG 1 TIME – Time of receiving broadcast message 1,

BROAD MSG 1 ID – Id of received message 1,

BROAD MSG 1 SENDER ID – Id of the message 1 sender,

⋮

BROAD MSG 4 TIME – Time of receiving broadcast message 4,

BROAD MSG 4 ID – Id of received message 4,

BROAD MSG 4 SENDER ID – Id of the message 4 sender.

Function of the ITEM module is evaluated from collected information. Current time slots of all devices are presented due to usage of Slot start packets. Using Inf packets will review correct slot time assignment and frame length setting. Remaining information is used to detect collisions of sent data.

The ITEM module behavior is evaluated depending on the placement of individual devices. JAVA application that visualizes the ITEM module function was developed for this purpose. We can easily prove correctness of the ITEM module function due to this application. Using of JAVA application is described in further text.

Applications for processing of collected data from the network are located in upper part of JAVA program. The ITEM module visualization application can be found on the tab called Item visualization (Figure 3.16).

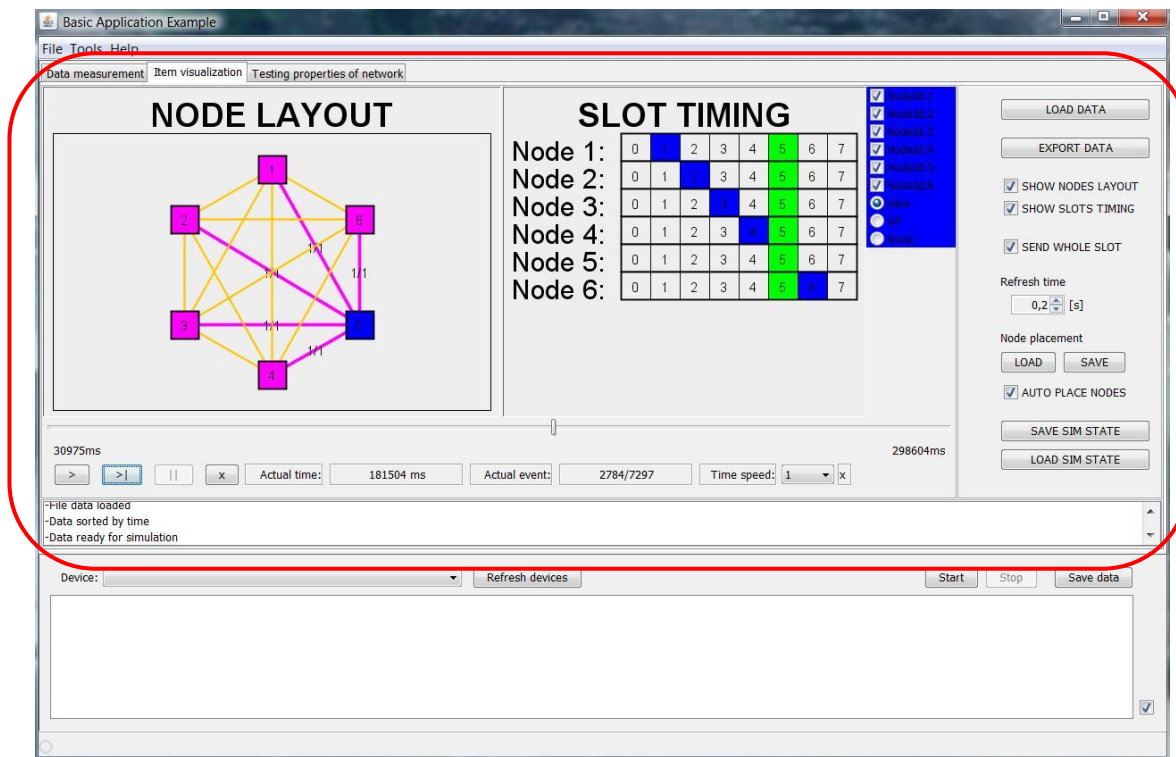


Figure 3.16 JAVA application for testing and visualization of ITEM function

The application is designed only for offline use. This means that visualized data must be already downloaded from the network and stored in a file. Offline use is based on used Collection module, which does not define order of received data. The information can be delivered with great delay. Offline use allows us to sort received information by time, because all pieces of information are available before simulation starts.

Application is divided into four parts. First part is visualization panel. Second part, options, serves for setting properties of simulation. Third and fourth part are timing and logging panel.

Visualization panel contains node layout panel, slot timing panel and list of participating nodes. We can turn on or off view of each device. We can simulate only requested part of the network with this option. Node layout panel shows layout of nodes in the network. As the device has no information about the current position, the position of a node on the layout panel must be set. Node positioning on the panel can be done manually (process is captured in Video folder on the DVD in file Application2Control.avi)

or automatically by the Auto place nodes property in options. Node positions can be saved in a file for future use.

Sending and receiving nodes are highlighted in node layout panel. Application evaluates simultaneous message reception. In case the device receives multiple messages at one time. Application signals an error. Sending of messages is so fast that user hardly records it. We can use stepping or set appropriately speed of simulation for this purpose.

We can choose to send the whole slot property in options. This property switches view of sending and receiving messages. Node layout panel highlights the nodes according to the slot assignment in this view. Connection between transmitting and receiving node shows number of transmitted and received messages in time slot.

Slot timing panel shows the sequence of generated slots. Application highlights current and assigned slots of each device in the network and signals an error in case there are conflict slots. Frame length is also displayed on this panel.

Simulation speed can be set on timing panel. Application can be launched with selected speed. We can use stepping on individual events. A slider moves actual simulation time. Actual state of simulation can be saved in a file.

Used information can be saved in a file by using export data button. We can export all or only selected information types.

This application can be also used in the case, when we want to verify actual node placement. When we place devices in space, some devices can be out of radio range. Otherwise, wrong placement can lead to communication between devices about which we thought they were out of radio range. We can verify actual node placement and make the necessary corrections with this application.

3.3.2 Testing of performance

After we verified the functionality we have to test the characteristics of the network with using the ITEM module. We evaluate the channel utilization and the time needed to transfer messages from source to destination (to the root of the collection tree). We have

chosen original Collection module (without the ITEM module) for comparison. We compare performance of TDMA versus CSMA protocol.

Proper application of TDMA protocol removes data packet collisions. On the other hand, using the E-ASAP protocol can reduce efficiency of channel utilization (reserved zero slot and guard periods in each slot in which we can't send data).

The testing principle is as follows. Each device in the network (except base station) generates messages with a given frequency and sends them to the root of the collection tree. Frequency is changed in a specified range. The root of the collection tree evaluates number of delivered messages and time needed to deliver. We have modified User application for data generating and evaluating. Connection of this application is consistent with the connection shown in Figure 3.10. A way of data generating in User application is presented in Figure 3.18.

Each device must have own assigned slot. Then we can start measurement by pressing the user button on each device. User application consists of two periods. The device generates and sends messages in first period. The second period serves to empty data queue (that stores the data for sending) among sending data periods. Sending data packet format is presented in Figure 3.17.

APP TYPE	NODE ID	TIME	MSG DELAY	MSG ID	MSG COUNT
1 B	2 B	4 B	2 B	2 B	2 B

Figure 3.17 Testing performance data packet (details sent between nodes)

Output meaning:

APP TYPE – Application Id (03 = testing performance of the network),

NODE ID – Node Id,

TIME – Message generation time,

MSG DELAY – Frequency of message generation,

MSG ID – Message Id,

MSG COUNT – Number of sent messages with a given frequency.

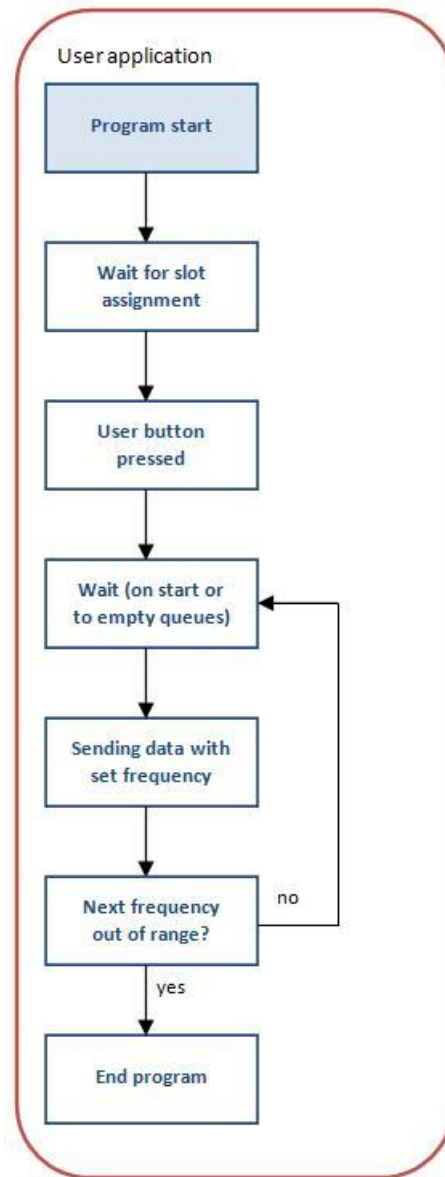


Figure 3.18 Testing performance data generation procedure (in case with the ITEM module)

The data packet contains message generation time. We can use local time of the ITEM module for this purpose.

The same User application can be used when testing performance of original Collection module (without the ITEM module). This application must be slightly modified. We do not need to wait for the assigned slot before running test. On the other hand, we do not have synchronized time from the ITEM module. We must create synchronized time. Principle of generating a synchronized time is as follows. One of the devices must be switched

on with User button. This device begins to generate its own local time. Local time is broadcasted into the network. The device that receives this time sets its own local time and it also begins broadcast local time into the network. This method of synchronization is not very accurate but it is sufficient for our purposes. When all devices in the network have synchronized time, we can start testing. A way of data generating in User application is shown in Figure 3.19.

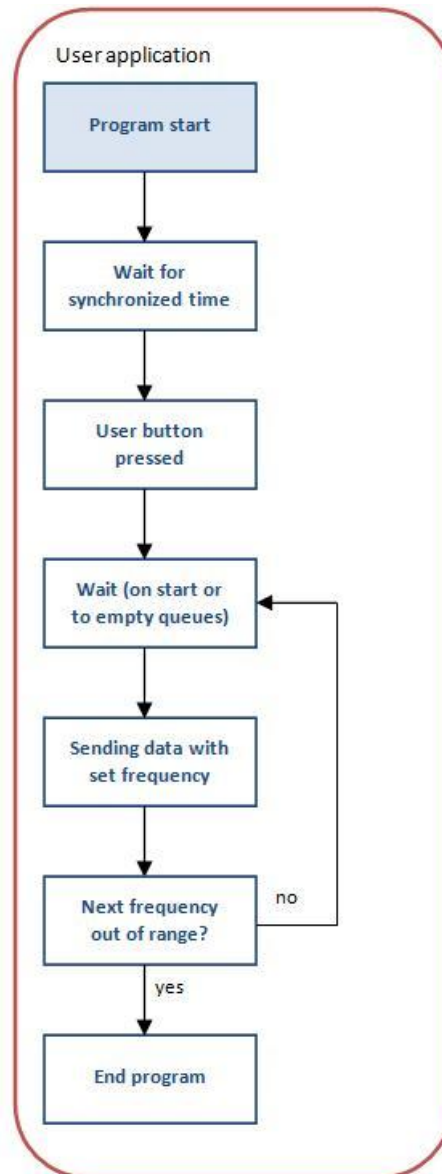


Figure 3.19 Testing performance data generation procedure (in case without the ITEM module)

The root of the collection tree collects data packets from the entire network and it creates statistics (number of received data packets, minimal, average and maximal time needed to deliver data packet to the root). These statistics are sent to the serial link of the connected computer for later processing and evaluation. Format of the data packet is same as the one presented in Figure 3.5. Item DATA in data packet is different and its format is presented in Figure 3.20.

APP TYPE	NODE ID	MSG DELAY	NR OF REC MSG	MSG COUNT	MIN DELAY	AV DELAY	MAX DELAY
1 B	2 B	2 B	2 B	2 B	4 B	4 B	4 B

Figure 3.20 Testing performance data packet (details sent to computer)

Output meaning:

APP TYPE – Application Id (03 = testing performance of the network),

NODE ID – Node Id,

MSG DELAY – Delay between generations of messages,

NR OF REC MSG – Number of received messages,

MSG COUNT – Number of sent messages with a given frequency,

MIN DELAY – Minimal time needed for data delivery,

AV DELAY – Average time needed for data delivery,

MAX DELAY – Maximum time needed for data delivery.

We have created application for a clear presentation of the results. This application can be found in upper part of JAVA program on tab called Testing properties of network as shown in Figure 3.21.

This application can be used online (reading data from serial link) or offline (read data from a file). The application consists of three parts. The main part occupies visualization panel that displays the measured statistics. Other parts are options and logging panel. The visualization panel consists of two graphs. First graph displays statistic of channel utilization (dependence of the number of received messages on the frequency of message

generation), second graph displays statistic of delivery duration (dependence of minimal, average or maximal time needed for data delivery on the frequency of message generation). There is a list of participating nodes in the next graphs. We can choose only certain devices to view and see requested part of the network. We can assign a color to each node for clear presentation of the results in graphs. Test results are dependent on the used placement of nodes in the network (nodes placed in lower layers of the collection tree give different result than nodes placed closer to the root of the tree).

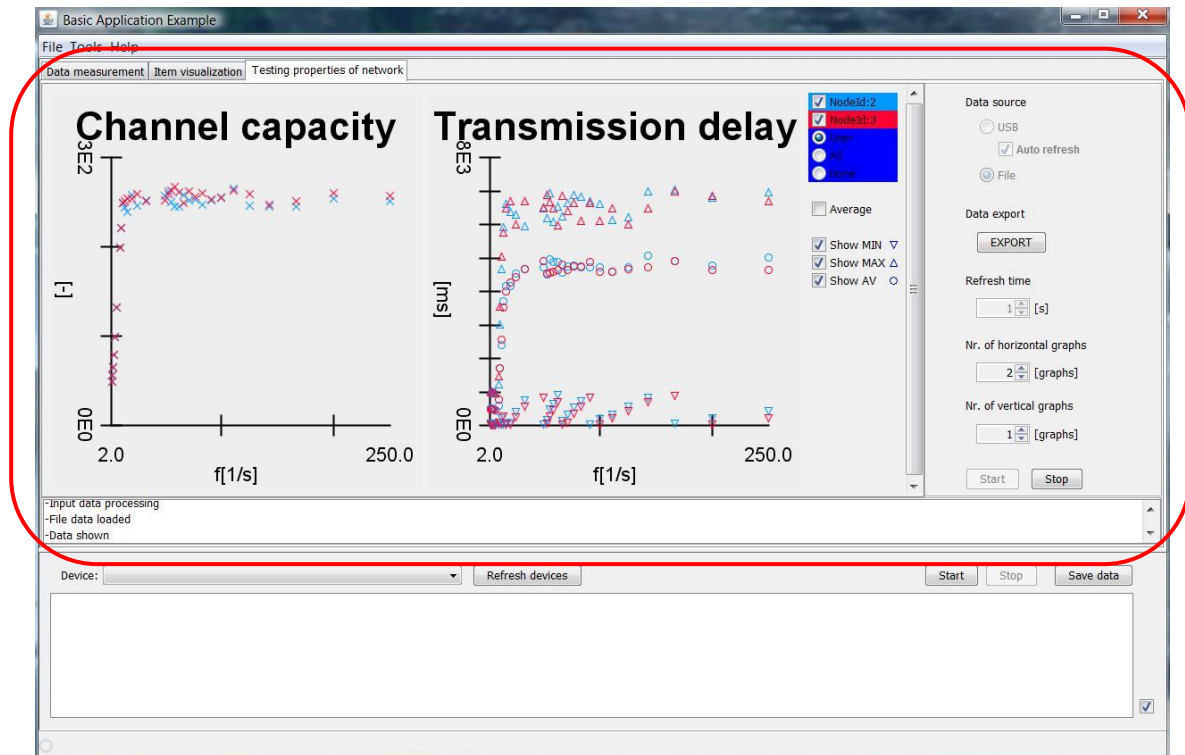


Figure 3.21 JAVA application for testing of ITEM performance

Measured data can be exported to a file in the same format as it is shown in Figure 3.20.

3.4 Data measurement application

Tracking of the required physical quantities in different locations in space is the most common application for the wireless sensor networks. Measured data must be delivered from the measuring point to the base station where it can be processed, evaluated and presented or recorded. We have created an application for this purpose.

The ITEM module ensures correct data transfer between neighbor nodes. The Collection module ensures delivering of measured data to the base station. Interconnection of these modules is shown in Figure 3.7. The implementation of data generation is in User application. The data are read from the sensor in the practical application. Nodes, that we used, did not have sensors. We have created a data generator that simulates the data obtained from sensors. Each generated data packet is forwarded to the Collection module for sending to the root of the tree. The root of the collection tree forwards received data to the serial link of the connected computer. This data packet format is the same as the one shown in Figure 3.5. Item DATA in data packet is different and its format is presented in Figure 3.22.

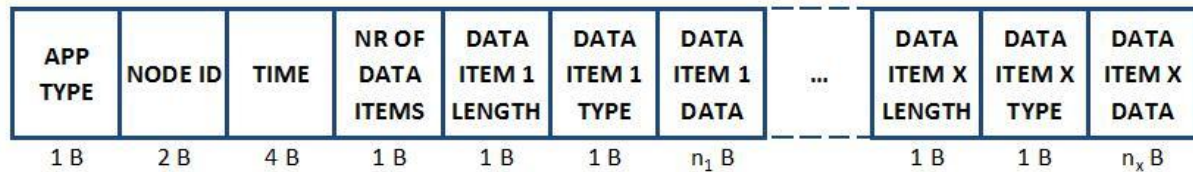


Figure 3.22 Data measurement packet details

Output meaning:

APP TYPE – Application Id (01 = data measurement),

NODE ID – Node Id,

TIME – Time of data packet generation,

NR OF DATA ITEMS – Quantity of generated data items in packet,

DATA ITEM X LENGTH – Length of data item X,

DATA ITEM X TYPE – Type of data item X,

DATA ITEM X DATA – Measuring of data item X.

Each data packet can contains needed quantity of measured data items due to the chosen data packet format. Each node can monitor different physical variables due to the distinction of data item types. Used Collection module does not guarantee delivery of data packets so the data packets can be lost. Use of the ITEM module reduces data loss by managing node access to the communication channel.

We have created application for clear presentation of measured data. This application can be found in upper part of JAVA program on tab called Data measurement as shown in Figure 3.23.



Figure 3.23 JAVA application for data measurement

We can use application online or offline as in the case of testing performance of network application. Frequency of data rendering can be set by choosing refresh time value in options (only for online application). Received data are displayed in the graph with corresponding type. We can assign a color to each node for clear presentation of the results in graphs.

We can see up to twelve graphs with different data types simultaneously. Graph axes are generated automatically according to measured values. We can use the zoom function for better view of details in the graph. Application control is captured in Video folder on the DVD in file Application1Control.avi.

Chapter 4

Experiments and results

We had to verify correct functionality of the modified ITEM module. First part of this chapter describes the results of practical tests carried out with the ITEM module. Application created for data measuring is tested in second part of this chapter.

The ITEM module behavior (slot assignment and frame length setting in EASAP module) is dependent on used nodes distribution in the network. Theoretically, we have an infinite number of ways to deploy nodes in space. We have a limited number of nodes to use in these tests. To achieve the required distribution of nodes we often need to have available space (even though we can set node's transmitting power and reduce node's radio range). For these reasons, we tested the cases that typically represent part of the network. Node with ID 1 represents the root of the Collection tree in all tests.

4.1 Testing and visualization of function

Three nodes participate in first test. Each node is in radio range within other nodes. The network distribution is shown in Figure 4.1. The testing is captured in Video folder on the DVD in file Test1_visualization.avi.

We can verify correctness of slot assignment during the simulation. We can see proper own slot assignment and corresponding frame length setting for all devices. The ITEM module function is correct and a data packet transmission is performed without collisions. We can also see a small example of frame length adjusting according to dynamic changes in the network. Visualization of this example is captured in Video folder on the DVD in file Test1a_frameReduction.avi. We start from the network distribution used in previous test. We can see corresponding frame length doubling when we add nodes with ID 4 and 5. When we remove these nodes from the network, we can observe correct frame length halving. Functionality of frame reduction when there are no nodes with slots in upper half of frame was successfully tested in original version. Tests of modified version of the frame reducing (frame

halving when possible with occupation of slots from upper half of frame) are shown in visualization. In the next step we add nodes with ID 4 and 5 again and then we remove nodes with assigned slots 2 and 3. We can see removing of nodes 4 and 5 in visualization. These nodes release slots from upper half of frame and wait with random delay for assignment of a new slot. After a while we can see assignment of slots from lower half of frame and corresponding frame reduction. Functionality of evaluating frame adjustment is correct.

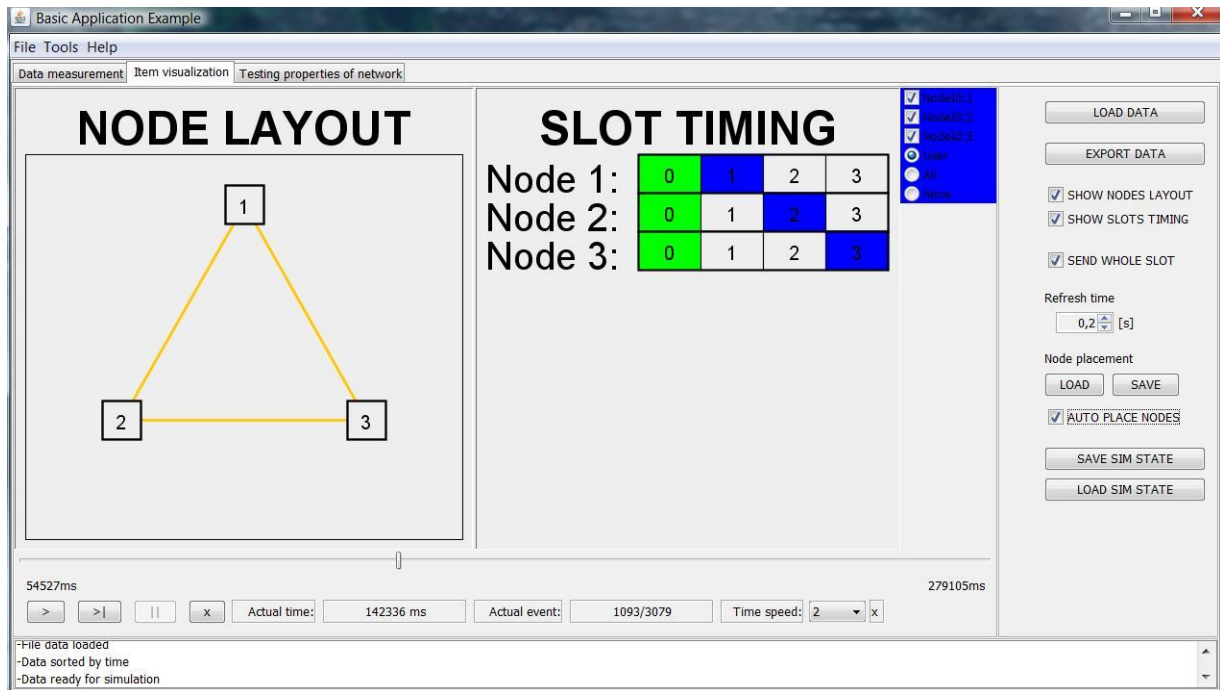


Figure 4.1 Test 1 network distribution

We study network performance for the case of three nodes in the network. Results of network with the ITEM module performance are presented in Figure 4.2. There are results of network without the ITEM module (original Collection module provided in TinyOS 2.x) for comparison in Figure 4.3.

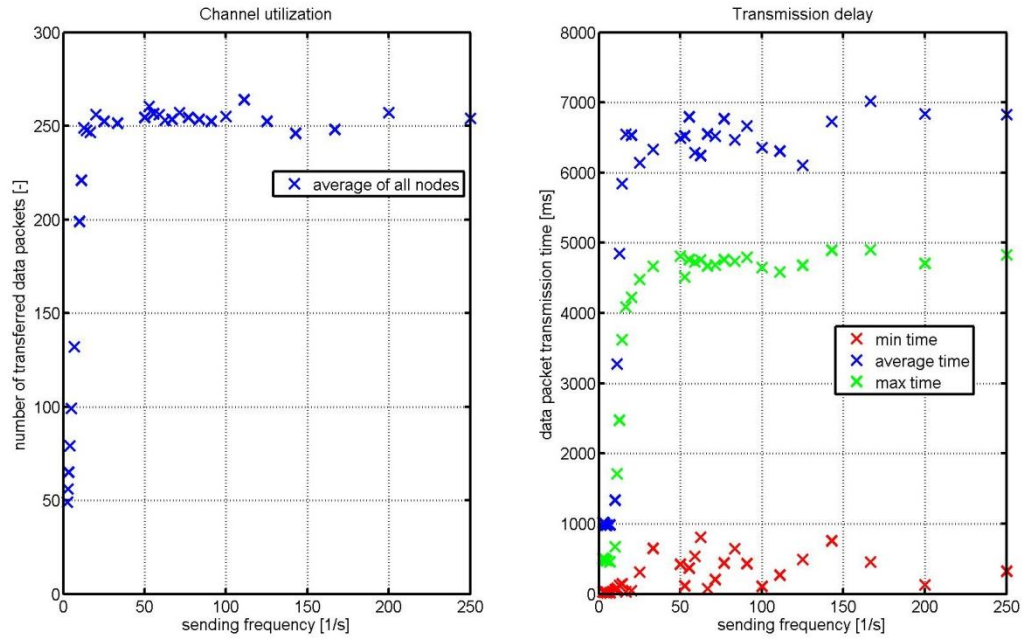


Figure 4.2 Test 1 network performance (network with the ITEM module)

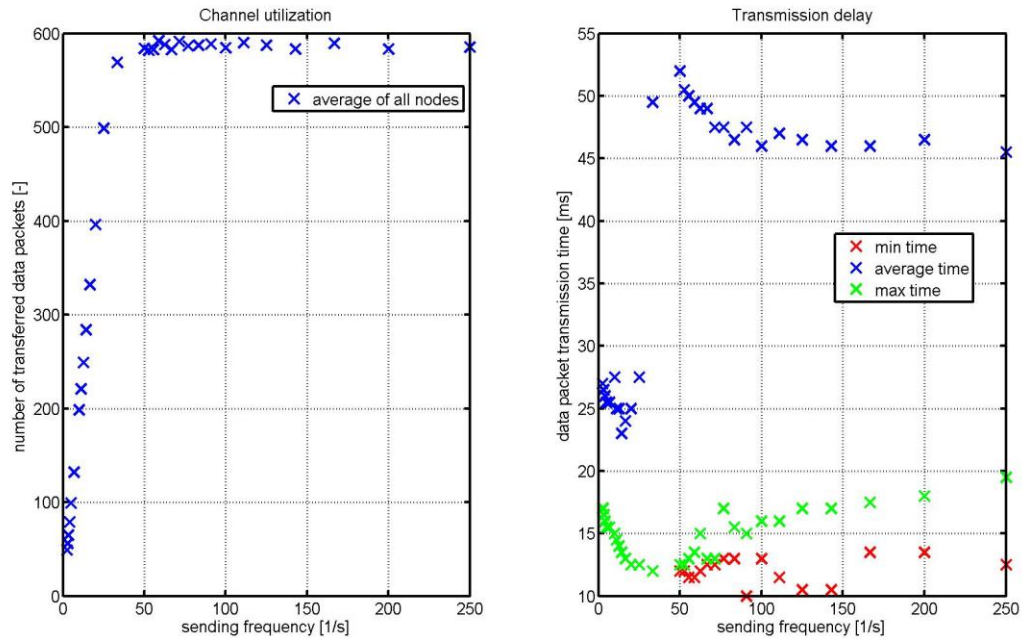


Figure 4.3 Test 1 network performance (original Collection module)

We can observe approximately half of the amount of data was transferred in network with the ITEM module. If we look at assigned slots in frame in the ITEM module, we can see that nodes use only two slots in frame length 4 (zero slot is reserved and first slot is occupied

by root of the tree, which only receives data). We expect worse results in measurement of data packet delivering time. Nodes must wait for own slot to send data. Time slot duration is 256ms in all tests. Average delivery time is worse by about two orders in the network with the ITEM module in comparison with network without the ITEM module.

Poor channel utilization affect the distribution of nodes in the network in another test. Each node is again within radio range of other nodes. We use seven nodes in this test. The network distribution is shown in Figure 4.4. Visualization of this test is captured in Video folder on the DVD in file Test2_visualization.avi.

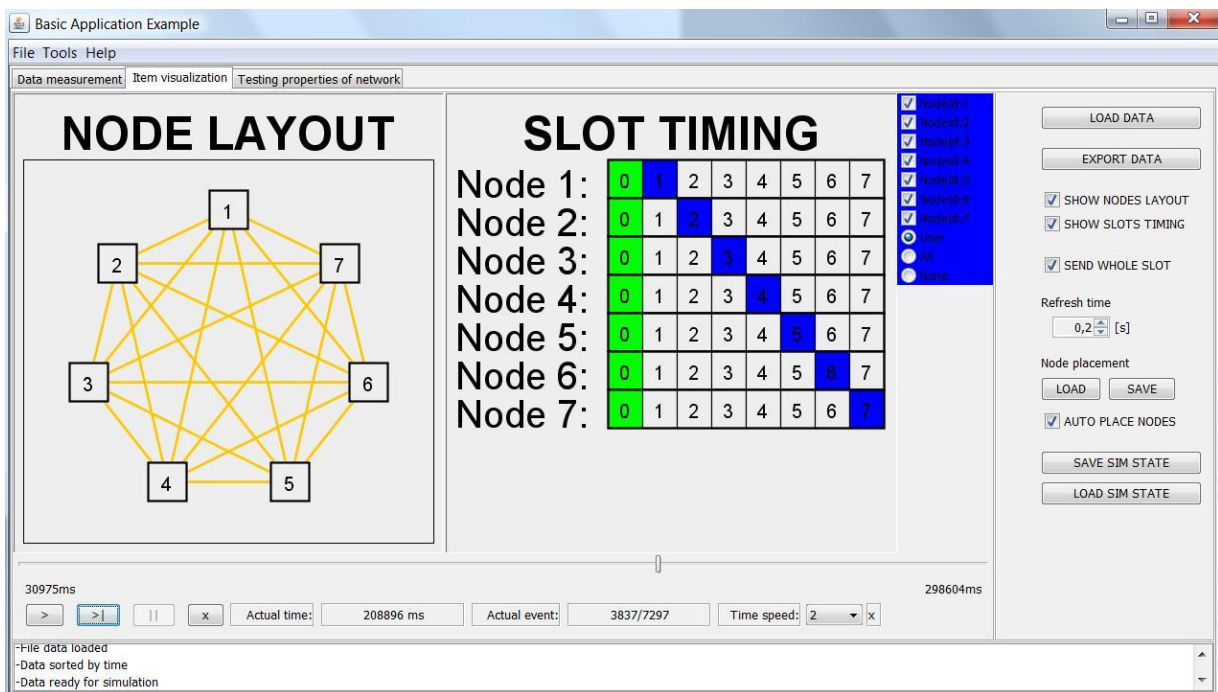


Figure 4.4 Test 2 network distribution

We can see correct slot assignment and frame length setting for all devices on the visualization. The ITEM module functionality is correct. Frame length of eight time slots is set. We guarantee better channel utilization in comparison with frame length of four slots in previous test. Only one quarter of frame is unused (zero and first slot) but the time between data packet transmitting is longer. Results are presented in Figure 4.5 (network with the ITEM module) and in Figure 4.6 (network without ITEM module).

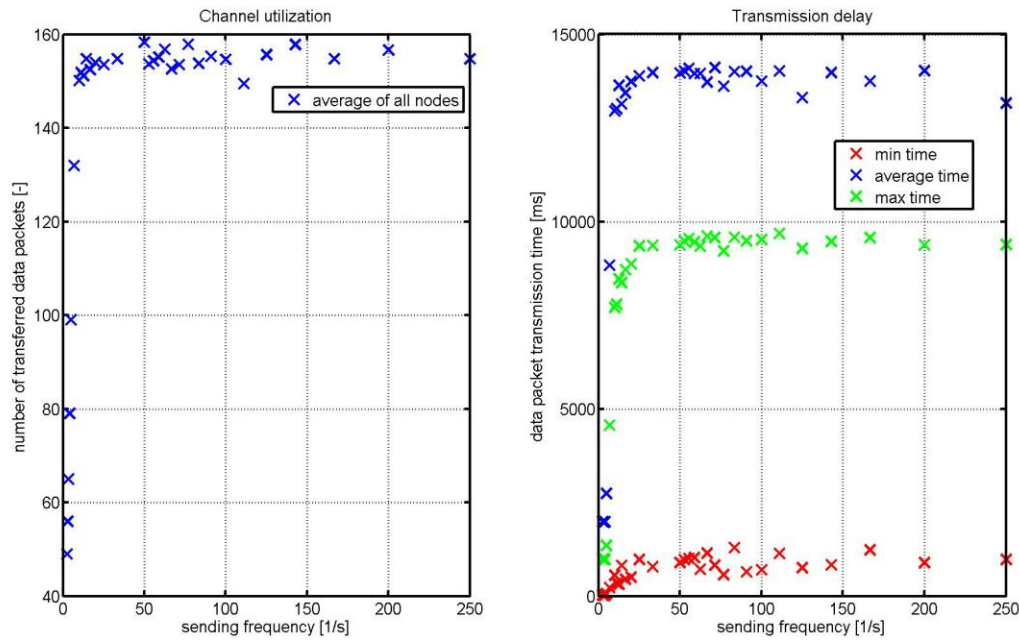


Figure 4.5 Test 2 network performance (network with the ITEM module)

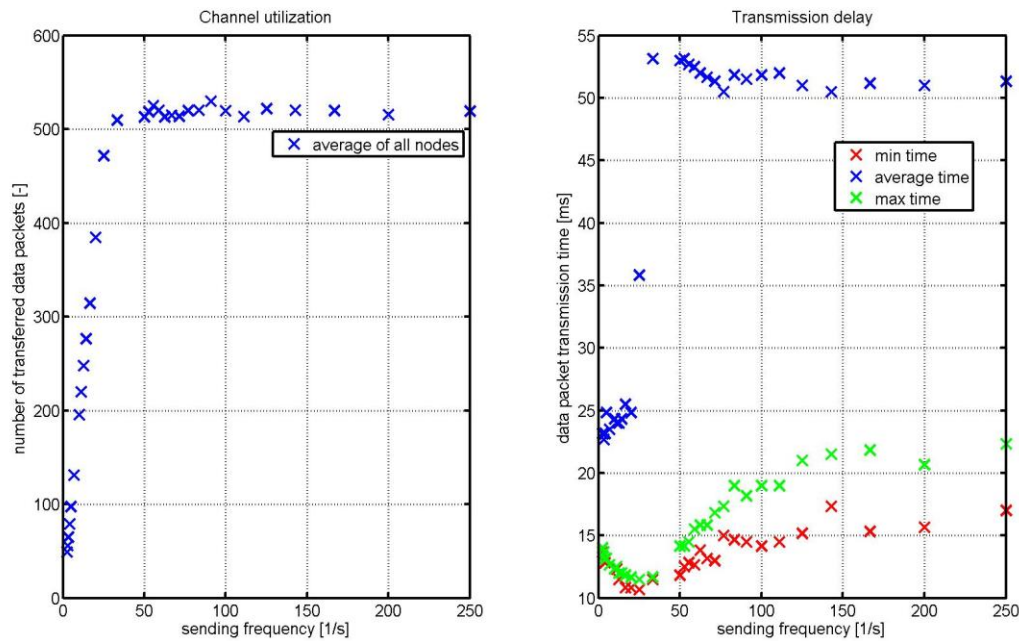


Figure 4.6 Test 2 network performance (original Collection module)

We can see approximately 3 times smaller amount of transferred data packets in the network with the ITEM module. The results of average delivery times are almost the same as in the previous test.

Distribution of nodes in the network in another test is shown in Figure 4.7. There are three nodes in this test. We try to maximize channel utility (all slot assigned) and short the period between own slots (minimal frame length).

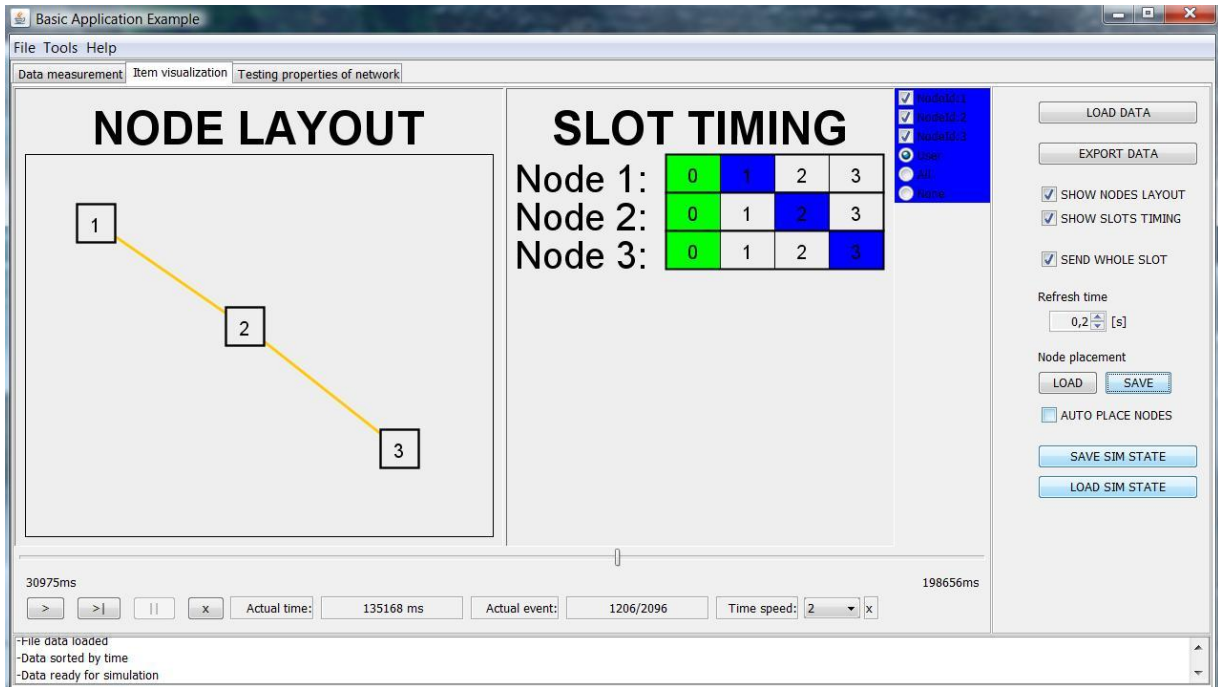


Figure 4.7 Test 3 network distribution

Visualization of this test is captured in Video folder on the DVD in file Test3_visualization.avi. Correct function of the ITEM module function can be observed in this visualization. Measured results of the network performance are shown in Figure 4.8 and Figure 4.9.

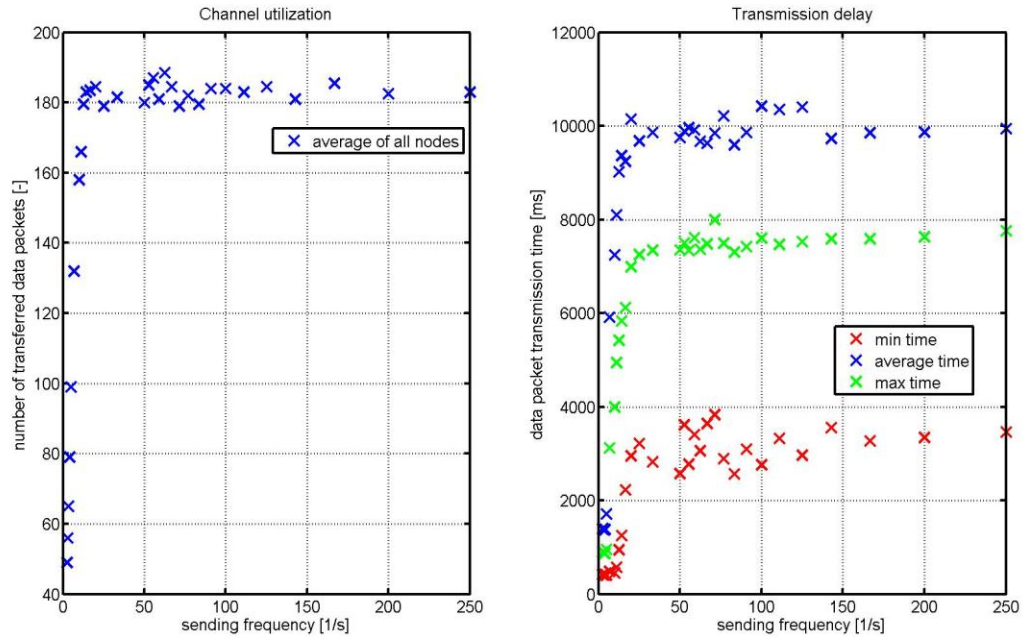


Figure 4.8 Test 3 network performance (network with the ITEM module)

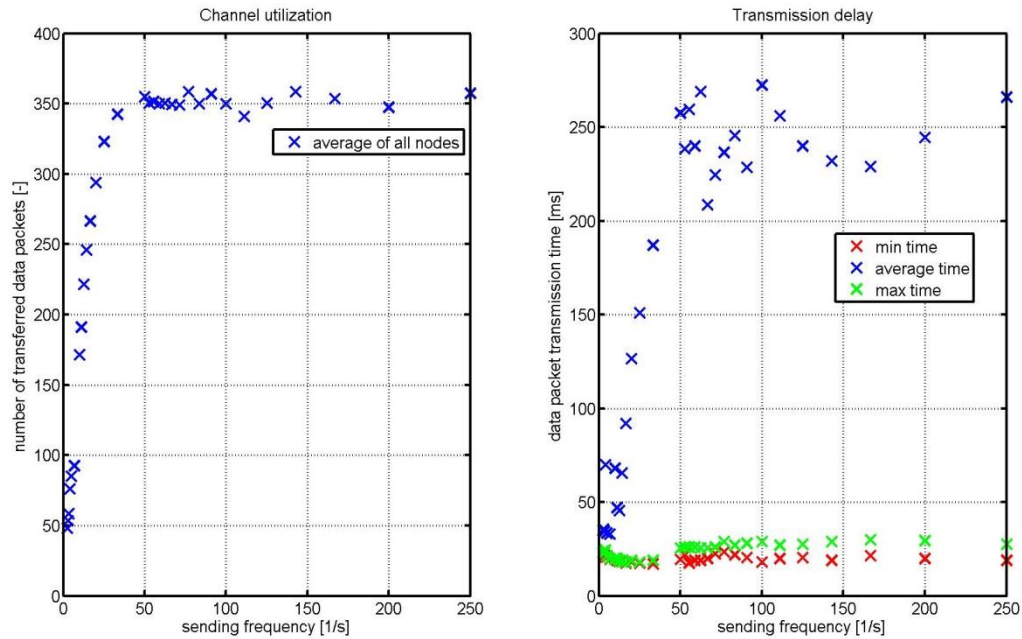


Figure 4.9 Test 3 network performance (original Collection module)

We can see that approximately half of amount of data was transferred in network with the ITEM module. This amount of transferred data packets is equivalent to quantity of assigned slots in frame. Results of average data packet transmission time are the same

as in previous tests. If we look at the performance of individual nodes (if we turn off averaging of results in application) we can see lower amount of transferred data packets from nodes that are closer to the root of the collection tree. This is due to Collection module implementation. Collection favors retransmitting of messages before sending of produced messages. We can see higher data packet transmission time at nodes at lower levels of the tree as expected.

We used five nodes in the last test. Their network distribution is presented in Figure 4.10. Nodes are deployed so that the frame length is four times slots length. Root of the collection tree is in the middle of other nodes. Visualization of this test is captured in Video folder on the DVD in file Test4_visualization.avi.

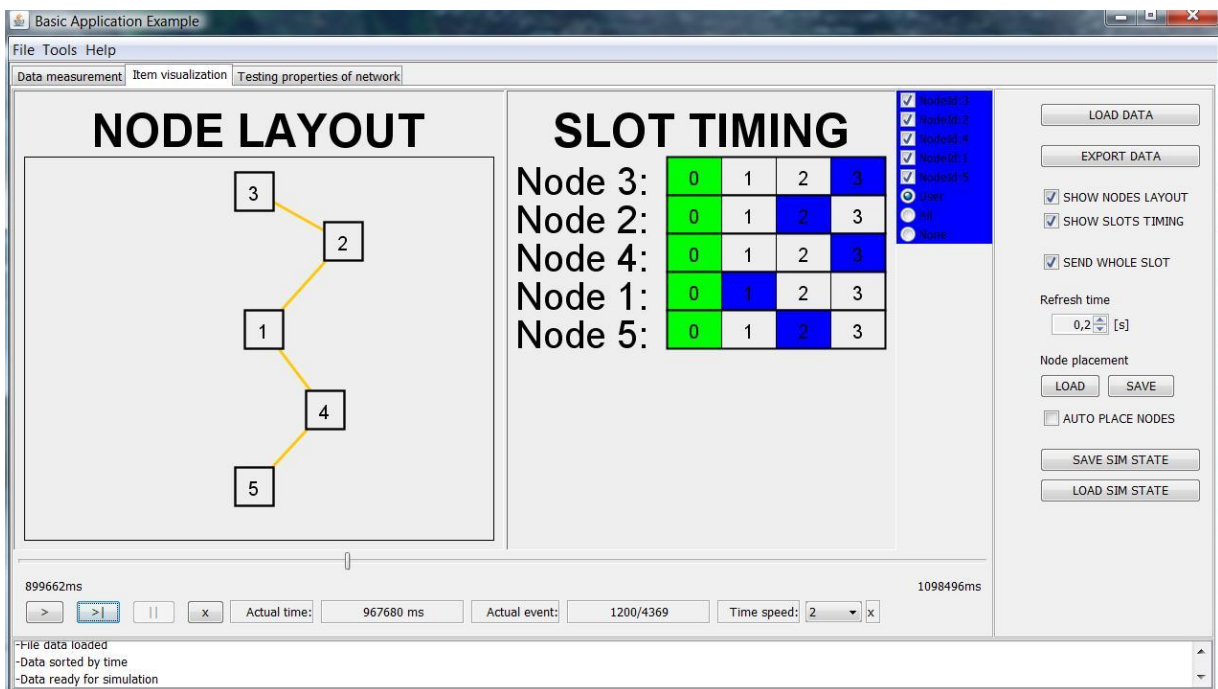


Figure 4.10 Test 4 network distribution

Results of the network performance are approximately the same as in previous tests. These results are shown in Figure 4.11 and in Figure 4.12.

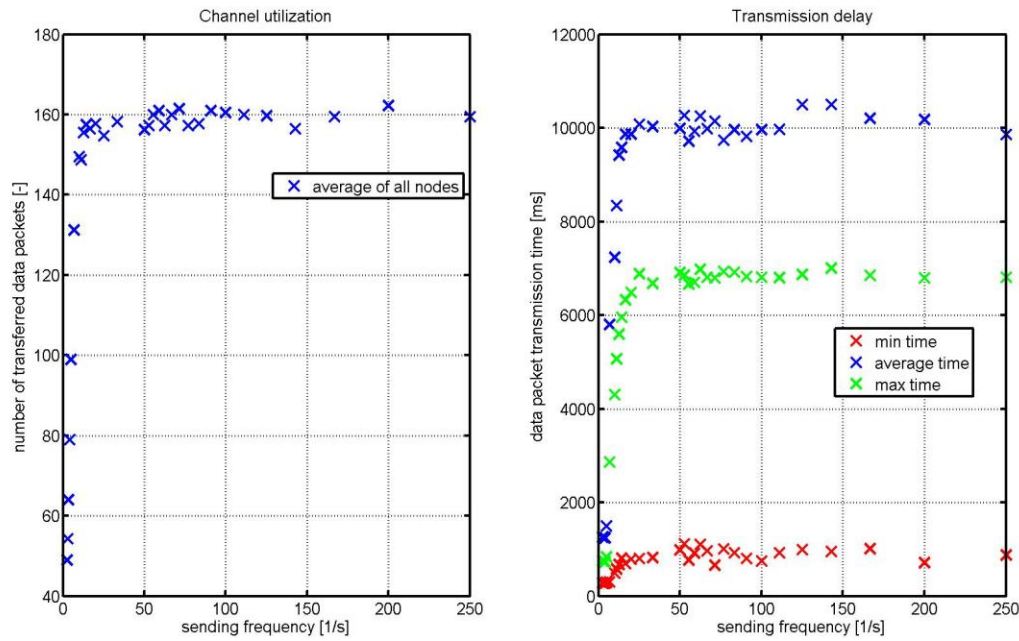


Figure 4.11 Test 4 network performance (network with the ITEM module)

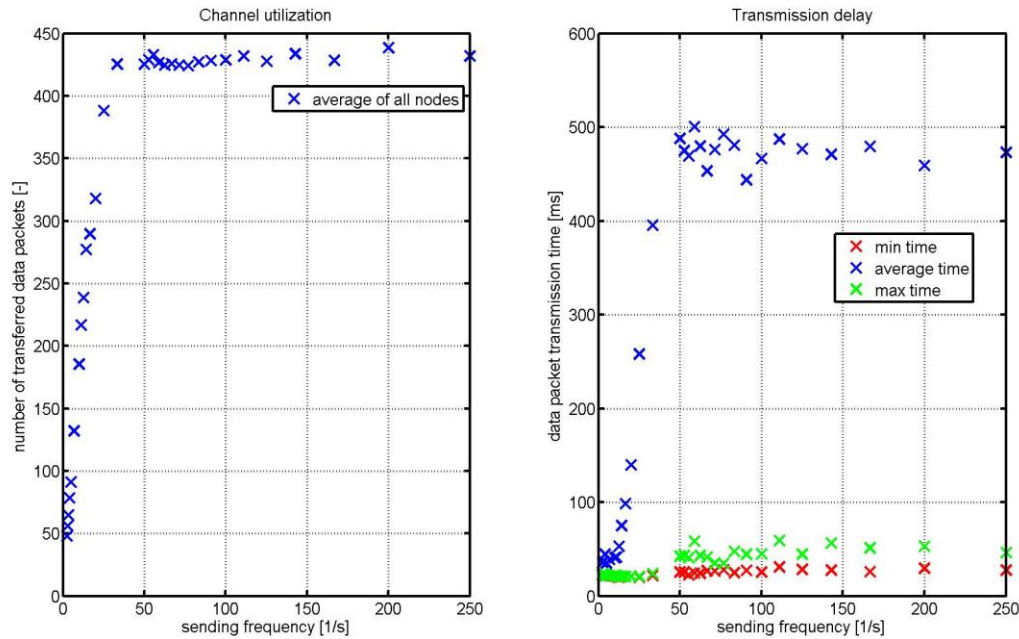


Figure 4.12 Test 4 network performance (original Collection module)

As we can see from performed experiments, usage of the ITEM module delivers worse results in the number of transmitted data packets and also in data packet transmission times in comparison with network without the ITEM module. These results are consistent

with theoretical expectations. Zero slot reserving, using of guard periods in TDMA protocol and also the way of evaluating the number of transmitted messages in own slot (the ITEM module compares remaining time in slot with time needed to send the longest message) reduces the possibility of using a channel capacity. A possibility to send messages only in node's own slot causes a significant increase of the time required to deliver a data packet. Under these assumptions we performed tests with a network distribution which will achieve the best possible results for network with the ITEM module (minimal possible frame length and all slot assigned).

Under these assumptions we performed tests with distribution of a network that achieves the best possible results for network with the ITEM module.

4.2 Testing of data measurement application

Functionality of application for data measurement is verified in this part. Nodes, used for testing, have no sensors to monitor various physical quantities so we have created simple value generator, which simulates measurement of real data.

We used three nodes in first test. All nodes generate the same type of data. This test simulates situation in which all nodes monitor the same physical quantities. Results can be shown in Figure 4.13.

We simulate situation in which each node monitor various physical quantity in second test. Results are presented in Figure 4.14.

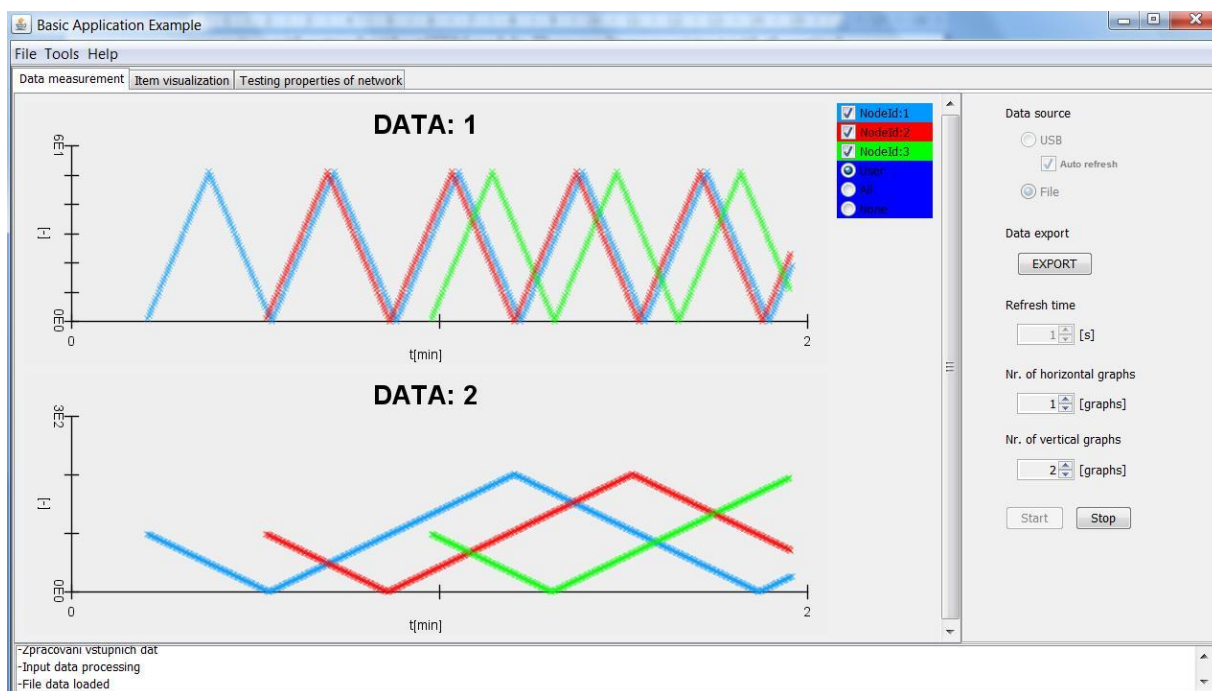


Figure 4.13 Test 1 application for data measurement

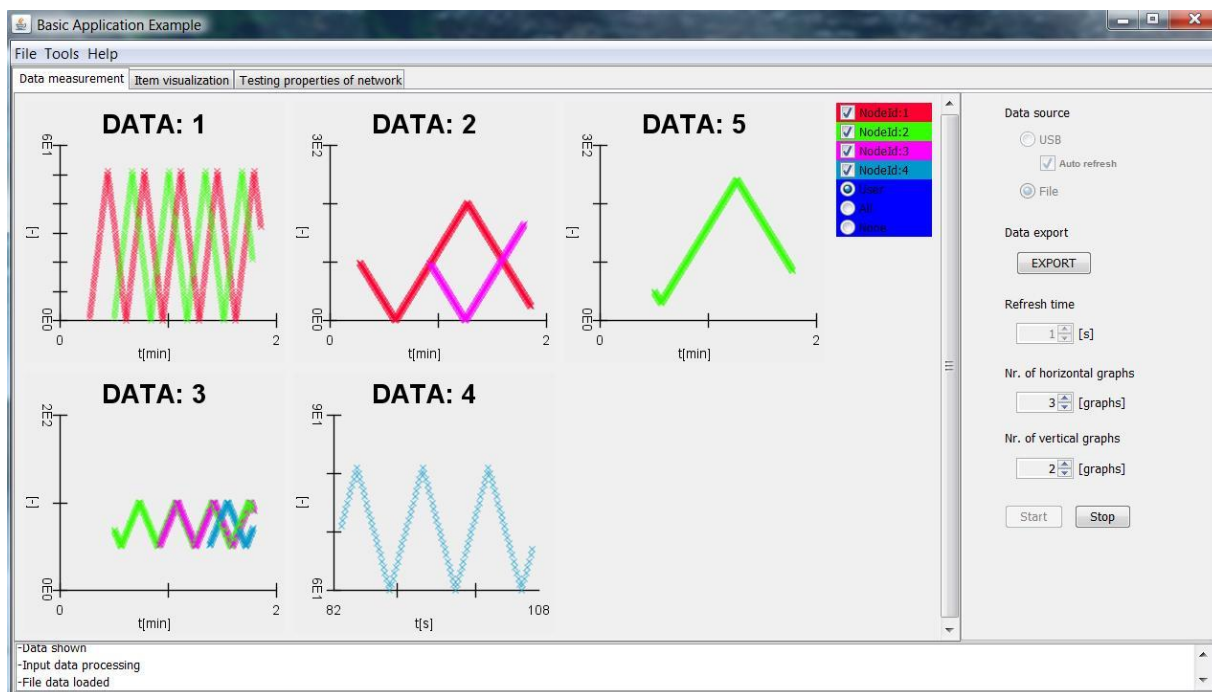


Figure 4.14 Test 2 application for data measurement

Chapter 5

Conclusion

The first part of the thesis was focused on testing and application of the software module for wireless sensor networks. Software module, called ITEM, was created for the purpose of controlling of wireless network communication.

We tried to improve the original version of this module. After elimination of some inadequacies and an adjustment for easier usage in practical applications a new version was created. This version had to be thoroughly tested before utilization in practical application.

As far as the network devices character is concerned, the principle of data collection and their re-evaluation was used. For the purpose of data collecting the Collection module provided in applications for TinyOS 2.x operating system was used. This module had to be modified for the correct interconnection to the ITEM module.

Later, we created two JAVA applications for test evaluation. The first application captures a sequence of events in the ITEM module. This visualization provides simple verification of this module. The second application serves for the evaluation of test of performance of network with the ITEM module.

Real tests of ITEM module were described in chapter 4. These tests proved correct function of the ITEM module. Performance of implemented TDMA protocol was compared with a performance of CSMA protocol which is used in the original Collection module. From the results of measurement we can observe smaller amount of transmitted data packets and higher data packet delivery times in TDMA protocol. In wireless sensor networks low energy consumption is important. Each node in the network exactly knows in which time slots it can transmit and receive data when it uses TDMA protocol. Out of these slots we can bring nodes to a state with low power consumption which prolongs their lifetime. The usage of CSMA protocol gives much higher power consumption in comparison with TDMA.

Finally, we have created the application for measurement of data in sensor networks. For easy evaluation of measured results was developed a JAVA application which clearly presents all data received from the network.

We still find some remaining inadequacies on the ITEM module. This topic is described in chapter 6. Elimination of inadequacies would boost the features of the ITEM module. Due to time restrictions, we were not able to eliminate all of them in this work. However, we think that this thesis helped to improve the ITEM module, which greatly simplifies application design for wireless sensor networks.

Chapter 6

Future work

This chapter describes possible improvements of the ITEM module, which were not implemented due to time restrictions.

The ITEM module was not tested with larger number of devices. Maximum tested frame length is eight. In the case of larger number of devices we need to split INF packet to multiple smaller parts and send it in multiple messages.

The possibility of reserving slots was not thoroughly tested.

Furthermore, it is necessary to solve the case where multiple user applications are connected to the ITEM module. There must be implemented an arbiter that divides the possibility of sending data among these applications.

Last of the possible modifications is implementation of putting a device to sleep in free time slots. This modification leads to save power consumption and prolong a lifetime of the device.

References

- [1] Horton, M. and Suh, J. A vision for wireless sensor networks. In 2005 IEEE MIT-S International Microwave Symposium Digest, 2005. 4 pp.
- [2] Cheng, M. and Li, D. Advances in Wireless Ad Hoc and Sensor Networks. 2008. ISBN 0-387-68565-0.
- [3] Levis, P. TinyOS Programming [online]. 2006 [rev. 2006-6-28], [cit. 2010-4-16].
<http://csl.stanford.edu/~pal/pubs/tinyos-programming.pdf>
- [4] Schenato, L. Wireless Sensor Network [online]. [cit. 2010-4-28].
<http://www.dei.unipd.it/~schenato/pics/SensorNetwork.jpg>
- [5] TelosB Data Sheet, Crossbow [online]. [cit. 2010-4-20].
http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf
- [6] MSP 430 F1611 Data Sheet, Texas Instruments [online]. 2009 [rev. 2009-3-18], [cit. 2010-4-20]. <http://focus.ti.com/lit/ds/symlink/msp430f1611.pdf>
- [7] CC2420 Data Sheet, Chipcon [online]. 2007 [rev. 2007-3-20], [cit. 2010-4-20].
<http://focus.ti.com/lit/ds/symlink/cc2420.pdf>
- [8] Tmote Sky Data Sheet, Moteiv [online]. 2006 [rev. 2006-11-13], [cit. 2010-4-20].
http://www.snm.ethz.ch/pub/uploads/Projects/tmote_sky_datasheet.pdf
- [9] Lamb, G. The TDMA Book. 1998. ISBN 0-966-57521-0.
- [10] Liu, J. ; Chiu, M. ; Lee, R. Communications Engineering: Essentials for Computer Scientists and Electrical Engineers. 2007.
- [11] Nor, S. M. Performance of CSMA-CA MAC protocol for distributed radio local area networks. In Proceedings 6th International Symposium on Personal, Indoor and Mobile Radio Communications, 1995. vol. 2, s. 912-916.

- [12] Du, D. H. C. ; Chang, S. P. ; Subbaro, G. Multiple packet multiple channel CSMA/CD protocols for local area networks. In Proceedings of the 8th Annual Joint Conference of the IEEE Computer and Communications Societies, 1989. vol. 1, s. 163-172.
- [13] Kanzaki, A. ; Hara, T. ; Nishio, S. An Adaptive TDMA Slot Assignment Protocol in Ad Hoc Sensor Networks. In Proceedings of the 2005 ACM Symposium on Applied Computing, 2005. S. 1160-1165.
- [14] Liu, R. P. ; et al. Efficient Reliable Data Collection in Wireless Sensor Networks. In Proceedings 68th Vehicular Technology Conference, 2008. s. 1-5.
- [15] Niannian Ding ; Xiaoping Liu, P. Data Gathering Communication in Wireless Sensor Networks Using Ant Colony Optimization. In Proceedings of the 2004 IEEE International Conference on Robotics and Biomimetics, 2004. S. 822-827.
- [16] Martinic, F. ; Schwiebert, L. Adaptive data collection in sensor networks. In Proceedings of 2009 2nd IFIP Wireless Days (WD), 2009. s. 1-6.
- [17] Fonseca, R. ; et al. Collection [online]. 2006 [cit. 2010-4-22].
<http://www.tinyos.net/tinyos-2.1.0/doc/html/tep119.html>
- [18] Singh, I. Real-Time Object Tracking with Wireless Sensor Networks, Diploma thesis. Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Control Engineering, 2007.
- [19] Beneš, P. ITEM, TinyOS module for sensor networks, Bachelor thesis. Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Control Engineering, 2008.

Contents of the DVD-ROM

The included DVD-ROM contains folders:

- **ITEM:** ITEM and testing modules.
- **Materials:** Materials used in thesis.
- **Testing application:** JAVA testing applications:
 - executable file in: Testing application/dist/TestingApplication.jar,
 - measured data in: Testing application/DataFolder,
 - documentation in: Testing application/dist/javadoc.
- **Thesis:** Diploma thesis in PDF format.
- **Video:** Demonstrations of testing application usage. Contain files:
 - Application1Control.avi – data measurement application control,
 - Application2Control.avi – testing and visualization of function application control,
 - Application3Control.avi – testing of properties application control,
 - Test1_visualization.avi – example of test1,
 - Test1a_frameReduction.avi – example of frame reduction,
 - Test2_visualization.avi – example of test 2,
 - Test3_visualization.avi – example of test 3,
 - Test4_visualization.avi – example of test 4.