# Testing application for
**for**
# **I**ntegrated **T**DMA **E**-ASAP **M**odule (ITEM)

**Version 1.0**

**Authors: Jiří Trdlička, Pavel Beneš**

**Czech Technical University**

**November 11, 2010**

# Testing application for ITEM

The 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. This document describes the principles of the testing methods and evaluation of the results. 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.

## 1. Testing and visualization of the ITEM 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 1.1.

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 presented in Figure 1.2. Item DATA in data packet is shown in Figure 1.3.

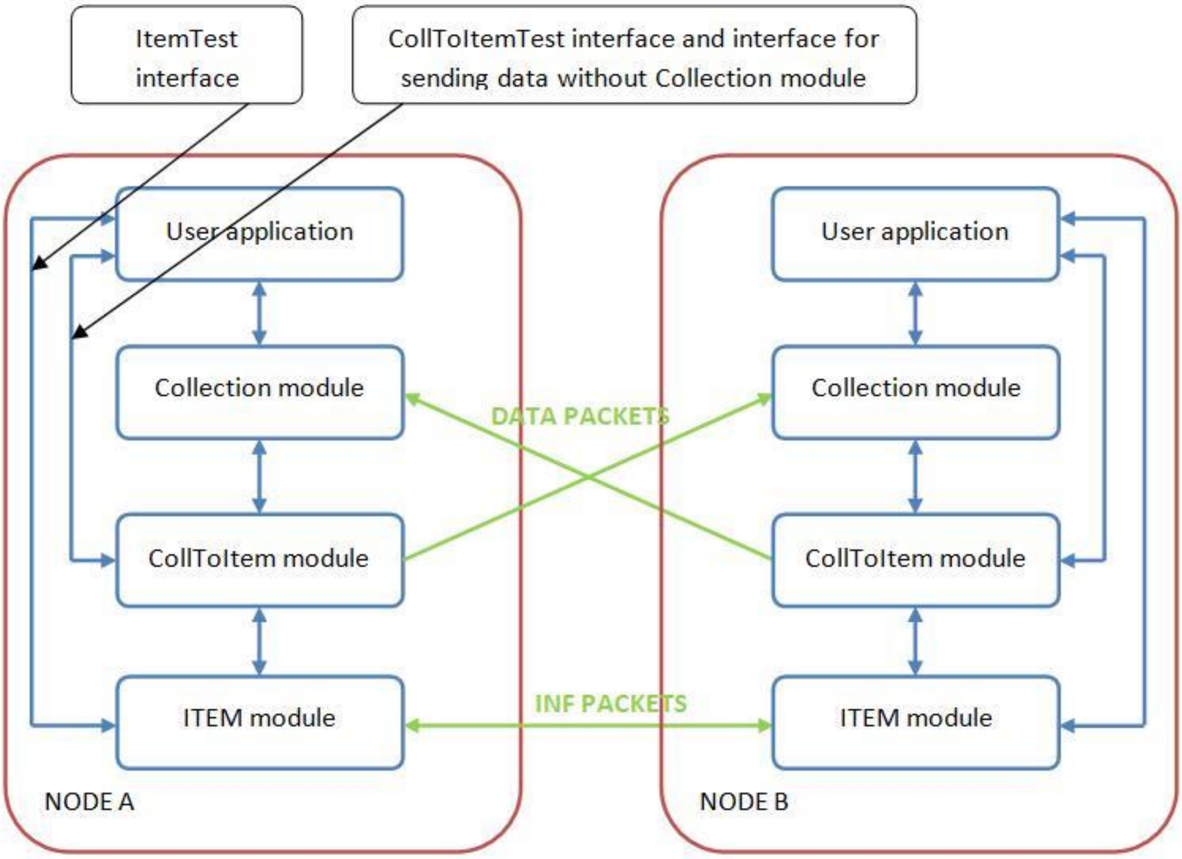


Figure 1.1 Interface for testing connection

| 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 1.2 General data packet format

| APP TYPE | NODE ID | TIME | INFORMATION |
|---|---|---|---|
| 1 B | 2 B | 4 B | n B |

Figure 1.3 ITEM visualization packet format

Output meaning:

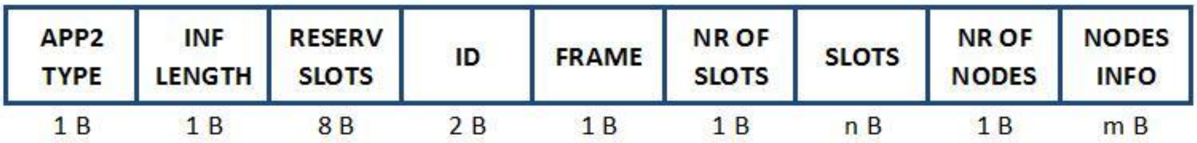
APP TYPE – Application id (02 = ITEM visualization),

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 1.4.



| APP2 TYPE | INF LENGTH | RESERV SLOTS | ID | FRAME | NR OF SLOTS | SLOTS | NR OF NODES | NODES INFO |
|---|---|---|---|---|---|---|---|---|
| 1 B | 1 B | 8 B | 2 B | 1 B | 1 B | n B | 1 B | m B |

Figure 1.4 ITEM visualization packet details – type Inf

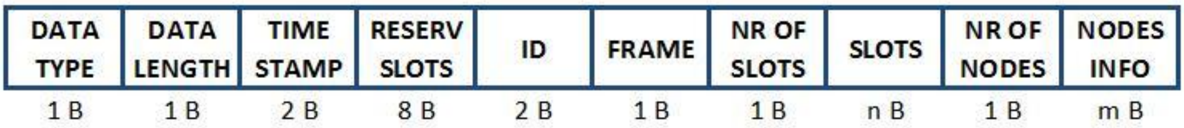
Output meaning:

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

INF LENGTH – Inf packet length,

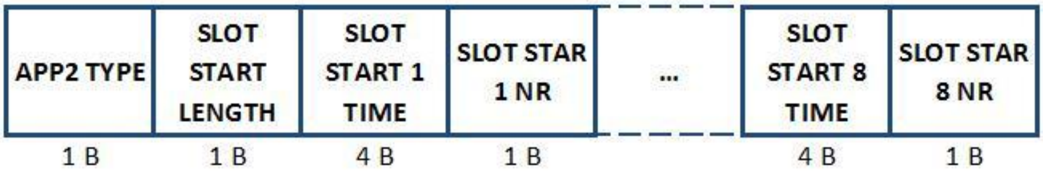
Rest of the data packet is a copy of real INF (copy starts from reserved slots item) packet used in the ITEM module (Figure 1.5).

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 1.6.



| 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 1.5 ITEM INF packet details



| APP2 TYPE | SLOT START LENGTH | SLOT START 1 TIME | SLOT STAR 1 NR | ... | SLOT START 8 TIME | SLOT STAR 8 NR |
|---|---|---|---|---|---|---|
| 1 B | 1 B | 4 B | 1 B |  | 4 B | 1 B |

Figure 1.6 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,

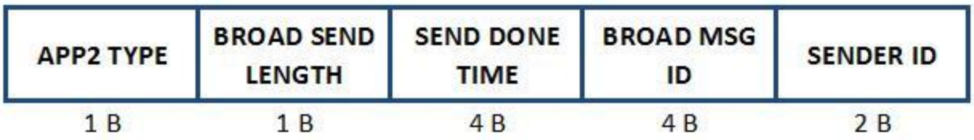
SLOT START 1 NR – Number of slot 1,

⋮

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 1.7.



| APP2 TYPE | BROAD SEND LENGTH | SEND DONE TIME | BROAD MSG ID | SENDER ID |
|---|---|---|---|---|
| 1 B | 1 B | 4 B | 4 B | 2 B |

Figure 1.7 ITEM visualization packet details – type Broadcast send

Output meaning:

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

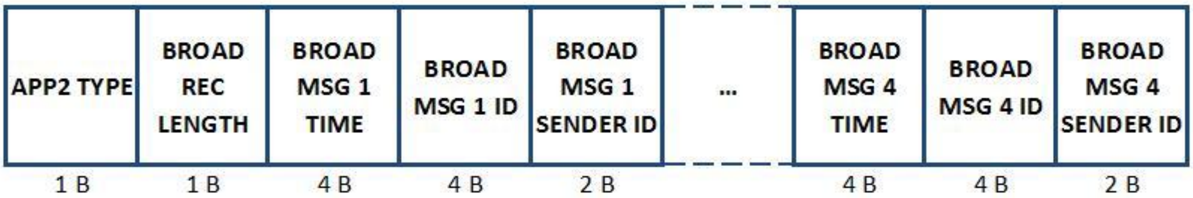
BROAD SEND LENGTH – Broadcast packet length,

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 1.8.



| APP2 TYPE | BROAD REC LENGTH | BROAD MSG 1 TIME | BROAD MSG 1 ID | BROAD MSG 1 SENDER ID | ... | BROAD MSG 4 TIME | BROAD MSG 4 ID | BROAD MSG 4 SENDER ID |
|---|---|---|---|---|---|---|---|---|
| 1 B | 1 B | 4 B | 4 B | 2 B | | 4 B | 4 B | 2 B |

Figure 1.8 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,

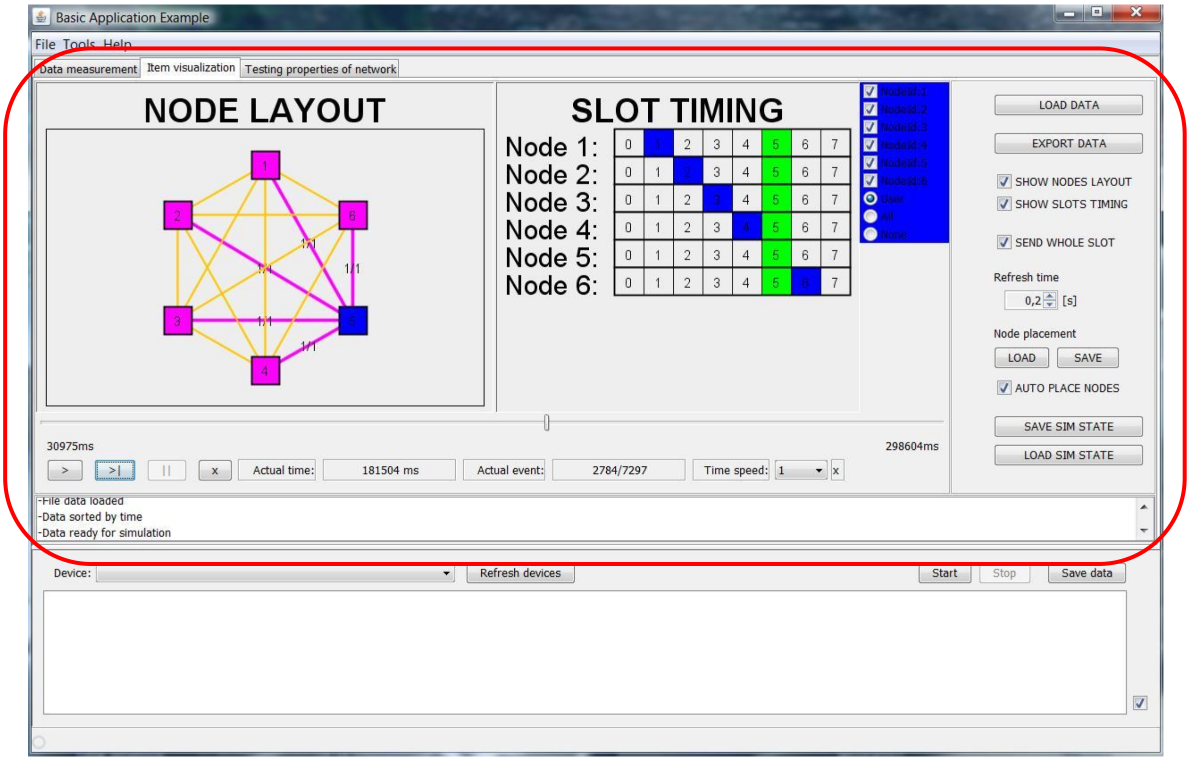
BROAD MSG 4 ID – Id of received message 4,

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 1.9).



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Node 1: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Node 2: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Node 3: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Node 4: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Node 5: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Node 6: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

☑ NodeId:1
☑ NodeId:2
☑ NodeId:3
☑ NodeId:4
☑ NodeId:5
☑ NodeId:6
● User
○ All
○ None

☑ SHOW NODES LAYOUT
☑ SHOW SLOTS TIMING
☑ SEND WHOLE SLOT

Refresh time 0,2 [s]

☑ AUTO PLACE NODES

Actual time: 181504 ms
Actual event: 2784/7297
Time speed: 1 x

```
-File data loaded
-Data sorted by time
-Data ready for simulation
```

Figure 1.9 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.

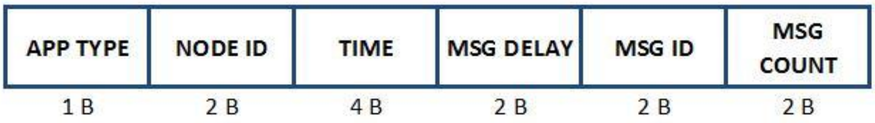
## 2. Testing of performance

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 1.1. A way of data generating in User application is presented in Figure 1.11.

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 1.10.



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

Figure 1.10 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.

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

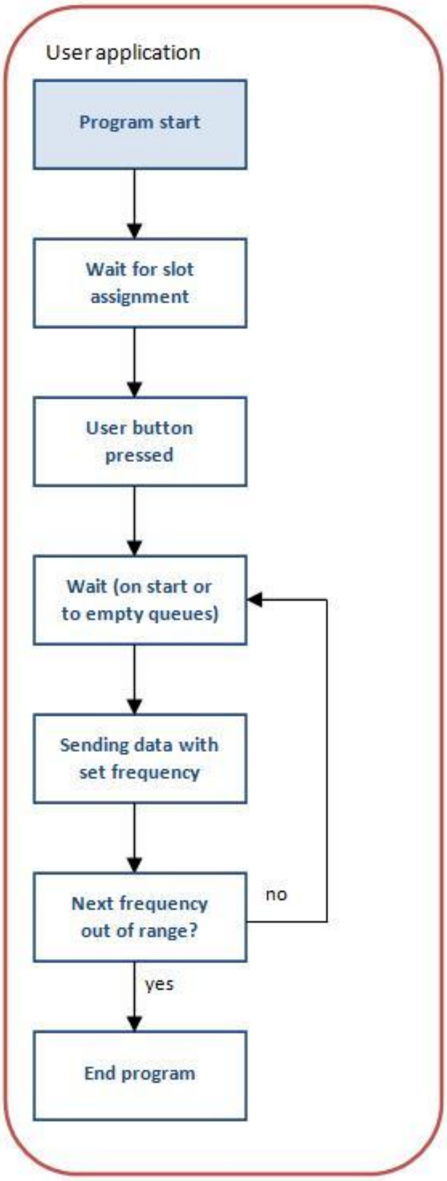

User application
Program start
Wait for slot assignment
User button pressed
Wait (on start or to empty queues)
Sending data with set frequency
Next frequency out of range?
no
yes
End program

Figure 1.11 Testing performance data generation procedure (in case with the ITEM module)

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 1.12.

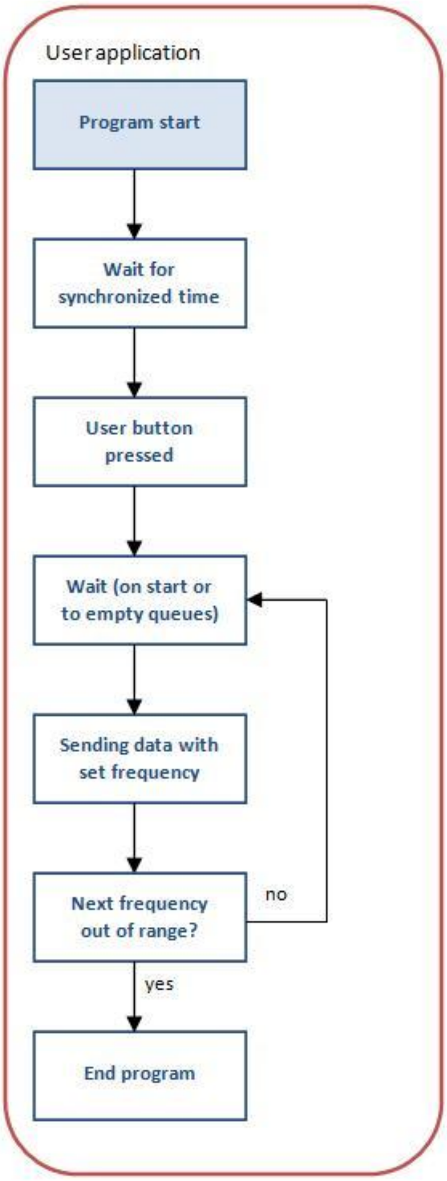
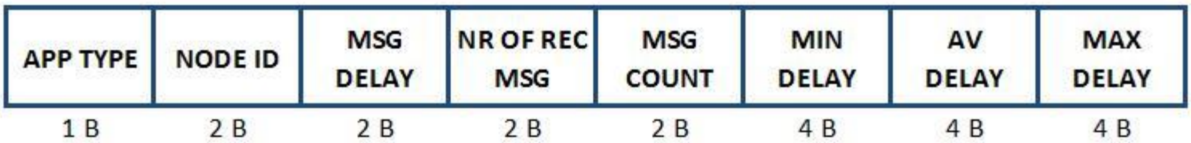

User application
Program start
Wait for synchronized time
User button pressed
Wait (on start or to empty queues)
Sending data with set frequency
Next frequency out of range?
no
yes
End program

Figure 1.12 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 1.2. Item DATA in data packet is different and its format is presented in Figure 1.13.



| 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 1.13 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,

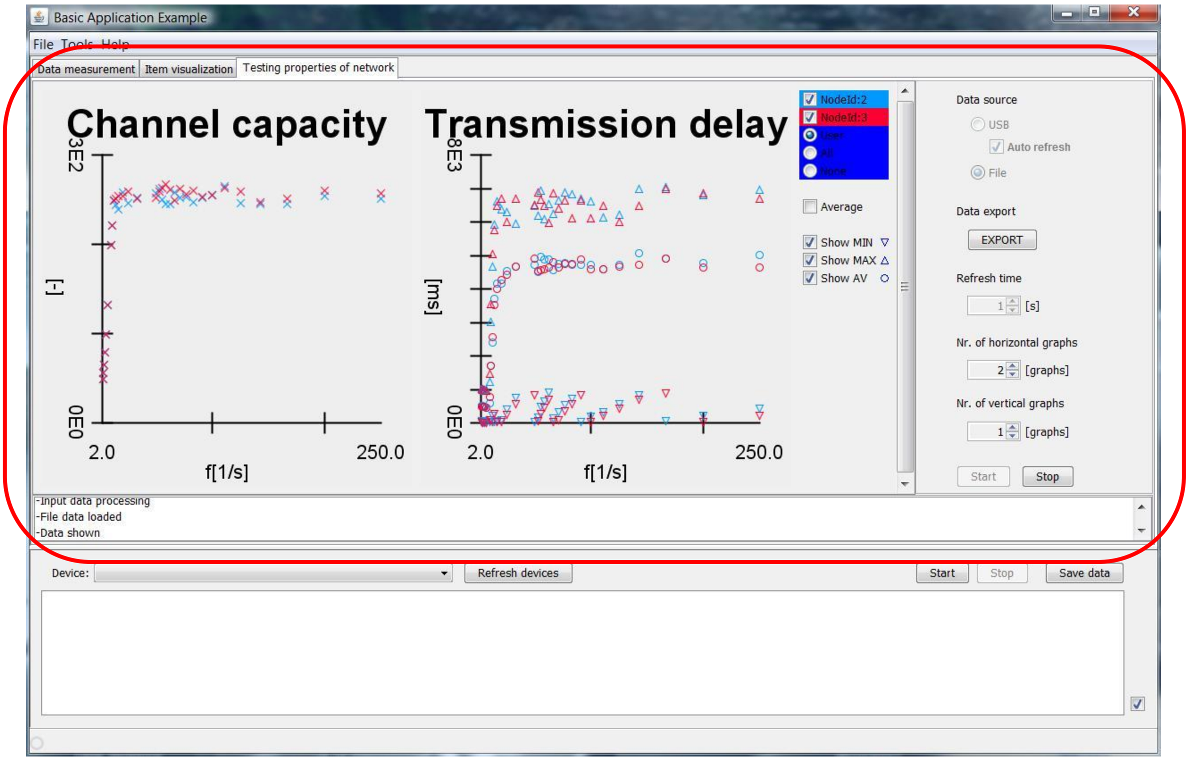
MIN DELAY – Minimal time needed for data delivery,

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 1.14.

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).



☑ NodeId:2
☑ NodeId:3
● User
○ All
○ None

□ Average

☑ Show MIN ▽
☑ Show MAX △
☑ Show AV ○

○ USB
☑ Auto refresh
● File

Refresh time: 1 [s]
Nr. of horizontal graphs: 2 [graphs]
Nr. of vertical graphs: 1 [graphs]

```
-Input data processing
-File data loaded
-Data shown
```

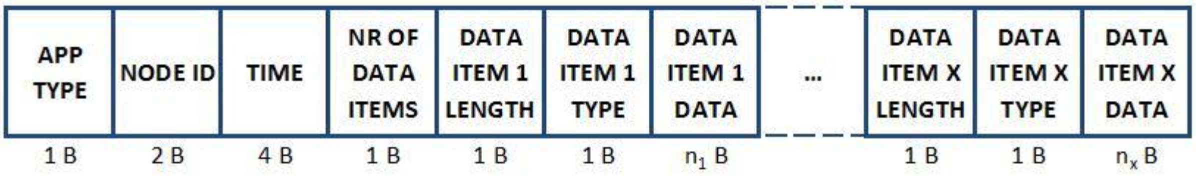
Figure 1.14 JAVA application for testing of ITEM performance

Measured data can be exported to a file in the same format as it is shown in Figure 1.13.

## 3. 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 1.1. 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 1.2. Item DATA in data packet is different and its format is presented in Figure 1.15.



| APP TYPE | NODE ID | TIME | NR OF DATA ITEMS | DATA ITEM 1 LENGTH | DATA ITEM 1 TYPE | DATA ITEM 1 DATA | ... | DATA ITEM X LENGTH | DATA ITEM X TYPE | DATA ITEM X DATA |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 B | 2 B | 4 B | 1 B | 1 B | 1 B | $n_1$ B | | 1 B | 1 B | $n_x$ B |

Figure 1.15 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,

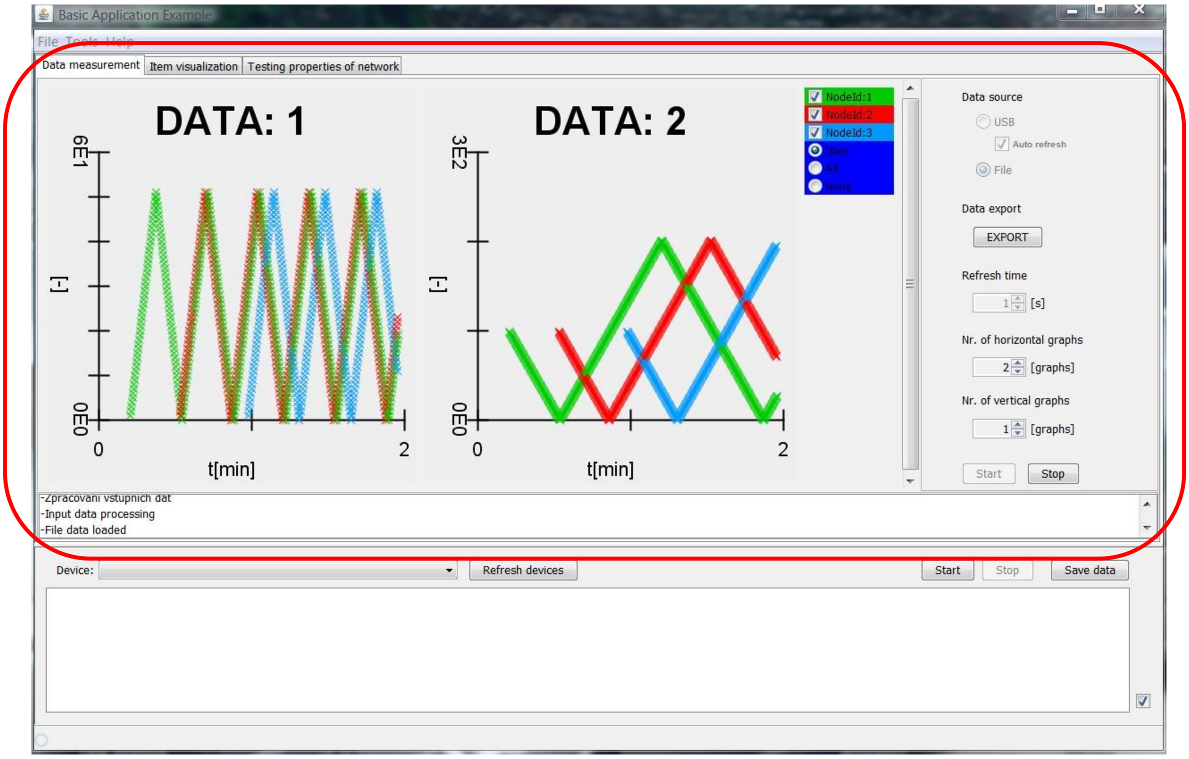
DATA ITEM X LENGTH – Length of data item X,

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 1.16.



☑ NodeId:1
☑ NodeId:2
☑ NodeId:3
● User
○ All
○ None

Data source
○ USB
☑ Auto refresh
● File

Data export

Refresh time: 1 [s]

Nr. of horizontal graphs: 2 [graphs]

Nr. of vertical graphs: 1 [graphs]

```
-Zpracovani vstupnich dat
-Input data processing
-File data loaded
```

Device:

Figure 1.16 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.