D5.7

Micro-ROS benchmarks - Extension

Grant agreement no.	780785
Project acronym	OFERA (micro-ROS)
Project full title	Open Framework for Embedded Robot Applications
Deliverable number	D5.7
Deliverable name	Micro-ROS benchmarks - Extension
Date	August 2021
Dissemination level	public
Workpackage and task	5.2
Author	Tomasz Kołcon (Łukasiewicz-PIAP), Alexandre Malki (Łukasiewicz-PIAP)
Contributors	Mateusz Maciaś (Łukasiewicz-PIAP), Andrzej Bratek (Łukasiewicz-PIAP)
Keywords	Microcontroller, NuttX, Benchmarking, Micro-ROS
Abstract	This deliverable summarizes works in Task 5.2: Benchmarking of whole stack.

Contents

1 Summary

This deliverable summarizes the works in Task 5.2: Benchmarking of whole stack. It is the fourth and one year extended iteration of benchmarking of the micro-ROS.

This document starts with explanations of the whole stack benchmarking methodology. Next, benchmarking setup will be presented. Subsequently, results are presented for each benchmark along with comments. At the end, conclusions are presented.

Term	Definition
ROS	Robot Operating System
MCB	Multi-connectors board
SWO	Single Wire Output
SWD	Serial Wire Debug
SWV	Serial Wire Viewer
JTAG	Joint Test Action Group (also name of interface)
ITM	Instrumentation (or Instruction) Trace Macrocell
ETB	Embedded Trace Buffer
OS	Operating System
RTOS	Real Time Operating System
HW	HardWare
IP	Internet Protocol
UDP	User Datagram Protocol
RAM	Random Access Memory
6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks.
I/O	Input/Output
ETM	Embedded Trace Macrocell
JSON	JavaScript Object Notation
UART	Universal Asynchronous Receiver-Transmitter
DDS-XRCE	DDS For Extremely Resource Constrained Environments
DDS	Data Distribution Service
KPI	Key Performance Indicators
CPU	Central Processing Unit
GCC	GNU Compiler Collection
PID	Process Identifier
CTF	Common Trace Format
SWOT	Strengths, Weaknesses, Opportunities, and Threats

2 Methodology

This deliverable is an update of D5.4 [1] for the 1-year extension of the project. The goal is to compare the results obtained for the Dashing with the Galactic version, which is the most up-to-date at the moment. All measurements were done in the same way like before and described in D5.4 [1].

3 Results

In deliverable D5.4 [1] all results of measurements for Dashing were collected on Github repository [2]. It was organized in this way due to the large amount of data. In addition the most of this data was difficult to analyze. For this reason, only selected data were presented along with an analysis and with the link to specific repository. This document is organized in the same way.

Complete results for Galactic version are available here:

https://github.com/micro-ROS/benchmarking-results/tree/master/aug2021/ [3]

3.1 Ethernet

3.1.1 General measurement at low-level communication medium and latency

Here results of the communication measurement are presented.

Complete results are available here:

- [3]/ether_publisher/com/20210817_17_12_1629213135_com_analysis.data [3].

After further analysis from the extracted data are as follow:

```
***********  TX Bandwith in kbps     **********

Avg for eth is 9066.716062127714
Max for eth is 47196.38490674594
Min for eth is 6033.816958128758

***********  RX Bandwith in kbps      ***********

Avg for eth is 14110.27107321563
Max for eth is 26041.666666666668
Min for eth is 13672.159836663264

***********  TX Latency  in microsec ***********

Avg for eth is 53.266110091743116
Max for eth is 55.81
Min for eth is 20.191

***********  RX Latency in microsec  ***********

Avg for eth is 33.34527678571428
Max for eth is 34.285
Min for eth is 18.0
```

According to the results, Galactic showed a lower minium latency. In addition the average TX bandwidth usage is smaller than in Dashing. However, the RX bandwidth usage is very similar to micro-ROS Dashing.

3.1.2 Real-Time and Determinism

Since both Dashing and Galactic use the same operating system, it makes no sense to make a Real-time and determinism comparison.

Results are available at [3]/ether_publisher/sched/hreadable [3].

3.1.3 Execution

Based on the same results taken from the scheduler the metrics were extracted. The software showed that most of the time the application is waiting for the packet to be sent away to the the application.

Complete results are available here:

- [3]/ether_publisher/sched/sched/20210816_21_32_1629142347_sched.json [3],
- [3]/ether_publisher/sched/hreadable.data [3].

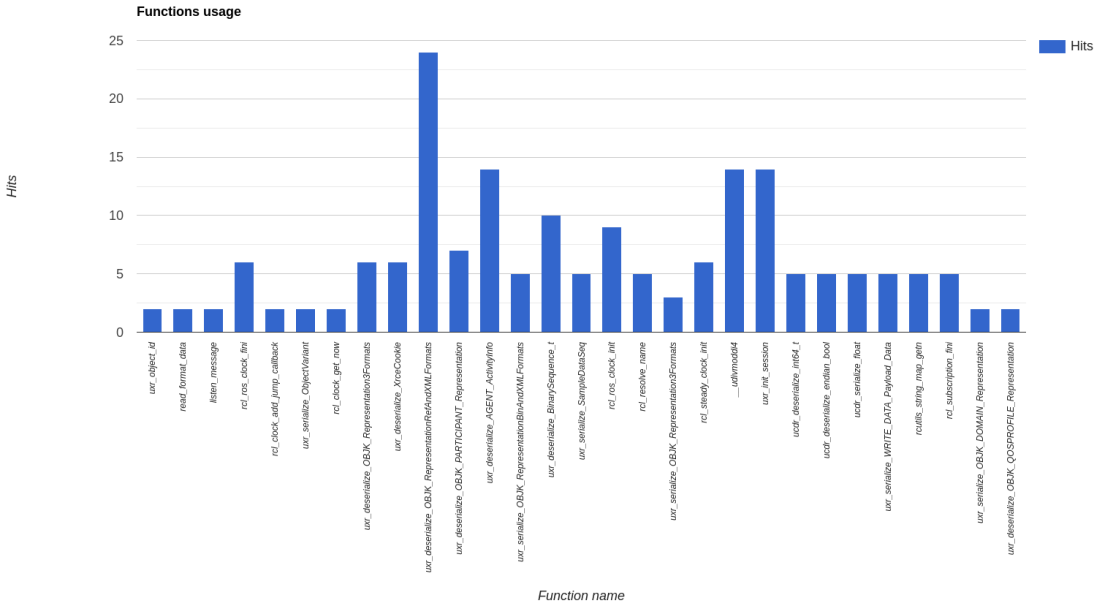
The timing execution are very comparable value as well. The amount of CPU taken by the microros application is around 0.23%. The most of the as in the Dashing, Galactic is waiting for I/Os.

3.1.4 Functions usage

Here the functions usage is presented.

Complete results are available here:

- [3]/ether_publisher/fusage/20200828_13_29_1598614159_fusage_analysis_hits.json [3],
- [3]/ether_publisher/fusage/hreadable.data [3].



The functions that are called the most often ($>$ 20 times) is:

- *uxr_deserialize_OBJK_RepresentationRefAndXMLFormats*: 23 times.

In Galactic, the number of function called during the publishing is less than in Dashing.

3.1.5 Static usage

At this point static memory usage is presented.

Complete results are available here:

- [3]/ether_publisher/mem-static/20210816_21_40_1629142826_mem_static.json [3],
- [3]/ether_publisher/mem-static/hdreadble.data [3].

Below the static memory usage results:

```
{"microros": {"stack_used_bytes": 440, "delim": 0}}
```

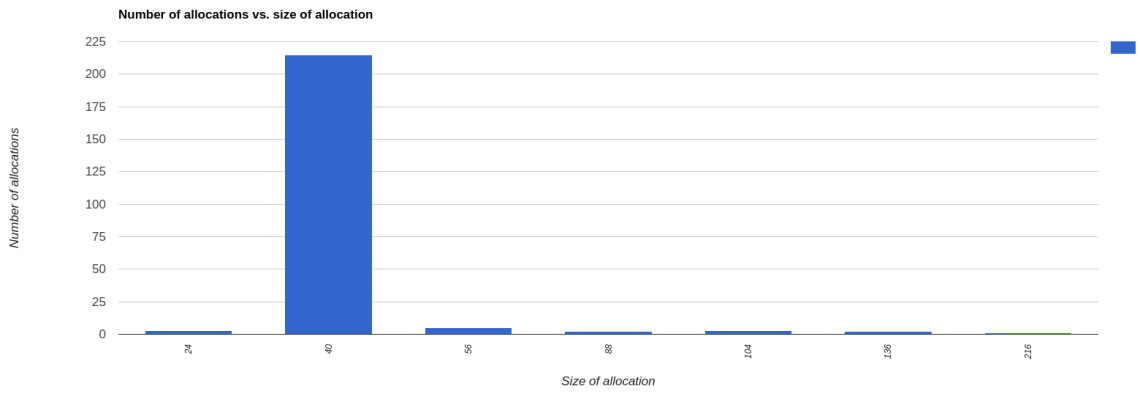
According to the results the amount of memory that the simple publisher is occupying is around 4400 bytes of stack memory. The memory usage in Galactic is reduced by a factor x20 in comparison to Dashing.

3.1.6 Heap allocation resources

Below are the results of dynamic memory usage measurements.

Complete results are available here:

- [3]/ether_publisher/mem-dyn/20210816_22_10_1629144602_mem_dyn.json [3],
- [3]/ether_publisher/mem-dyn/hreadable.data [3].



Regarding the heap allocation results, it's noticeable that in comparison the number of dynamic allocation is greater on Galactic than on Dashing. However the total size of allocation is Galactic is lower than in Dashing.

3.1.7 Power usage

The results are available here:

- [3]/ether_publisher/pwr/measure_info.txt [3],
- [3]/ether_publisher/pwr/pwr_oscilloscope.PNG [3].

The power usage is around 579mW. The power usage in Galactic is smaller than in Dashing.

3.1.8 Interrupts and interrupt service routines

As the same equipment was used with the same RTOS, the measurements regarding the ISR are leading to the same results. Therefore they would be worthless.

3.1.9 Communication resilience

After running the different tests, the communication resilience results of micro-ROS bring similar results as the previous deliverable. In a nuttshell, when the connection is failing or is unstable, the micro-ROS on NuttX does not provide a failsafe solution.

More details regarding the result can be found here with the output of the agent and the output of the client.

- [3]/ether_publisher/com_resilience/publisher_delay_1000ms.txt [3]
- [3]/ether_publisher/com_resilience/microros_delay_5000ms.txt [3]
- [3]/ether_publisher/com_resilience/microros_lost_10_pourcents.txt [3]
- [3]/ether_publisher/com_resilience/microros_lost_60_pourcents.txt [3]
- [3]/ether_publisher/com_resilience/microros_currupted_1_pourcent.txt [3],
- [3]/ether_publisher/com_resilience/microros_currupted_10_pourcents.txt [3].

Additionally the packet captures are also available:

- [3]/ether_publisher/com_resilience/microros_delay_1000ms.pcapng [3]
- [3]/ether_publisher/com_resilience/microros_delay_5000ms.pcapng [3]
- [3]/ether_publisher/com_resilience/microros_lost_10_pourcents.pcapng [3]
- [3]/ether_publisher/com_resilience/microros_lost_60_pourcents.pcapng [3]
- [3]/ether_publisher/com_resilience/microros_currupted_1_pourcent.pcapng [3],
- [3]/ether_publisher/com_resilience/microros_currupted_10_pourcents.pcapng [3].

3.2 Serial

3.2.1 General measurement at low-level communication medium and latency

Here results of communication measurement are presented.

Complete results are available here:

- [3]/serial_publisher/com/20210811_21_18_1628709505_com_analysis.data [3]

After further analysis from the relevant extracted data are as follow:

```
***********  TX Bandwith in kbps     **********
Avg for uart is 59.04314703568811
Max for uart is 73.57095771729918
Min for uart is 37.28583019138071

***********  RX Bandwith in kbps      ***********
Avg for uart is 59.706575676073456
Max for uart is 60.31886967263743
Min for uart is 58.176334797825604
```

```
***********  TX Latency  in microsec ***********
Avg for uart is 13.359920265780731
Max for uart is 21.238
Min for uart is 12.762

***********  RX Latency in microsec  ***********
Avg for uart is 13.085183006535948
Max for uart is 13.429
Min for uart is 12.952
```

The value for the communication are very similar for Galactic and Dashing.

3.2.2 Real-Time and Determinism

Since both Dashing and Galactic use the same operating system, it makes no sense to make a Real-time and determinism comparison.

Results are available at

- [3]/serial_publisher/sched/20210816_17_06_1629126415_sched.json [3],
- [3]/serial_publisher/sched/hreadable [3].

3.2.3 Execution

Here are the results of program execution time during communication cycle.

Complete results are available here:

- [3]/serial_publisher/sched/20210816_17_06_1629126415_sched.json [3],
- [3]/serial_publisher/sched/hreadable [3].

The results in Galaxy are very similar to Dashing. Indeed the most of the time, the application is waiting for the I/O to perform transfers.

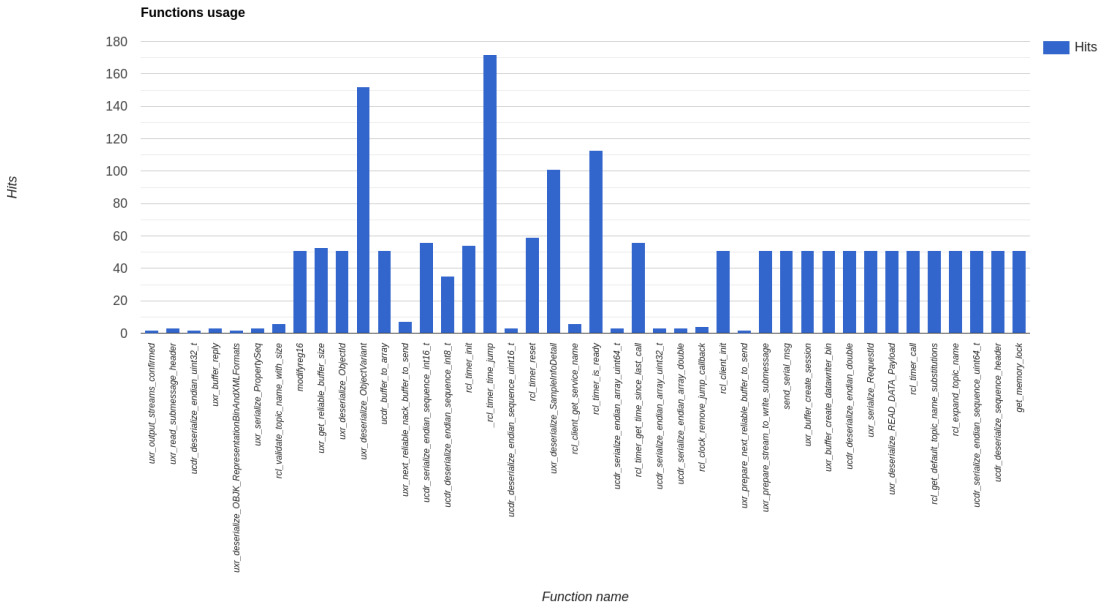
3.2.4 Functions usage

Here the functions usage is presented.

Complete results are available here:

- [3]/serial_publisher/fusage/20210812_19_45_1628790303_fusage_analysis.json [3],
- [3]/serial_publisher/fusage/hreadable.data [3].

After extraction of the information from the generated trace the data are plotted as follow:



This picture is showing that the most of the time is spent in the functions:

- _rlc_timer_time_jump: 166 times
- uxr_deserialize_ObjectId: 146 times,
- rlc_timer_is_ready uxr_deserialize_SampleInfoDetail: 100 times.

In general in the Galactic version has a smaller count of function call than in the Dashing version.

3.2.5 Static usage

At this point static memory usage is presented.

Complete results are available here:

- [3]/serial_publisher/mem-static/20210812_18_26_1628785614_mem_static.json [3],
- [3]/serial_publisher/mem-static/hreadable.data [3].

Below the static memory usage results:

```
{"microros": {"stack_used_bytes": 3224, "delim": 0}}
```

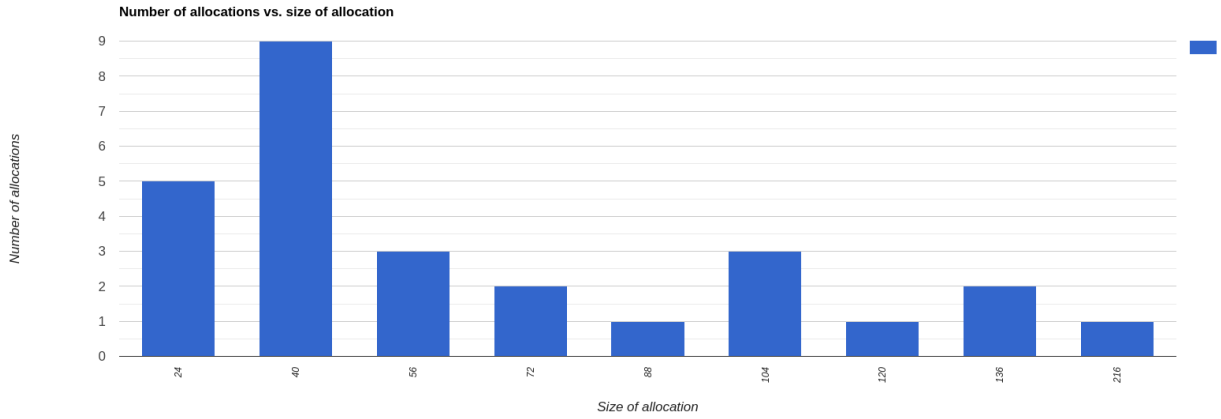
The Galactic version of micro-ROS has reduce the amount of stack usage by more than 4 tims in comparison to the micro-ROS Dashing.

3.2.6 Heap allocation resources

Below are the results of dynamic memory usage measurements.

Complete results are available here:

- [3]/serial_publisher/mem-dyn/20210811_21_44_1628711040_mem_dyn.json [3],
- [3]/serial_publisher/mem-dyn/hreadable.data [3].



The dynamic memory allocation is also improved in the Galactic version, in comparison to the Dashing version.

3.2.7 Power usage

The results are available here:

- [3]/serial_publisher/pwr/measure_info.txt [3],
- [3]/serial_publisher/pwr/pwr_oscilloscope.PNG [3].

The power usage is around 437mW. The power usage is very similar in Galactic and Dashing.

3.2.8 Interrupts and interrupt service routines

As the same equipment was used with the same RTOS, the measurements regarding the ISR are leading to the same results results. Therefore they would be worthless.

3.3 6LoWPAN

It should be noted that communication measurement results may be unpredictable due to the nature of radio communication.

3.3.1 General measurement at low-level communication medium and latency

Here results of communication measurements are presented. The results for the 6LoWPAN are different the measurements only show the SPI transfer speed between the CPU and the MRF24J40.

Complete results are available here:

- [3]/6lowpan_publisher/com/20210821_13_30_1629545441_com_analysis.data [3]

```
***********  TX Bandwith in kbps     **********
Avg for radio is 15.591879374858078
Max for radio is 339.1052272586346
Min for radio is 7.444699284772851
```

```
***********  RX Bandwith in kbps      ***********
Avg for radio is 76.03947984782864
Max for radio is 77.48275406078506
Min for radio is 75.08924893587808

***********  TX Latency in microsec ***********
Avg for radio is 1300.827290909091
Max for radio is 1679.047
Min for radio is 137.238

***********  RX Latency in microsec  ***********
Avg for radio is 494.0917454545455
Max for radio is 544.476
Min for radio is 493.047
```

As shown in the results above, the Galactic version is using less bandwidth than the Dashing version.

3.3.2 Real-Time and Determinism

Since both Dashing and Galactic use the same operating system, it makes no sense to make a Real-time and determinism comparison.

Results are available at:

- [3]/6lowpan_publisher/sched/20210821_13_11_1629544300_sched.json [3],
- [3]/6lowpan_publisher/sched/hreadable [3].

3.3.3 Execution

Based on the same results taken from the scheduler the metrics were extracted. The software showed that most of the time the application is waiting for the packet to be sent away to the the application.

Complete results are available here:

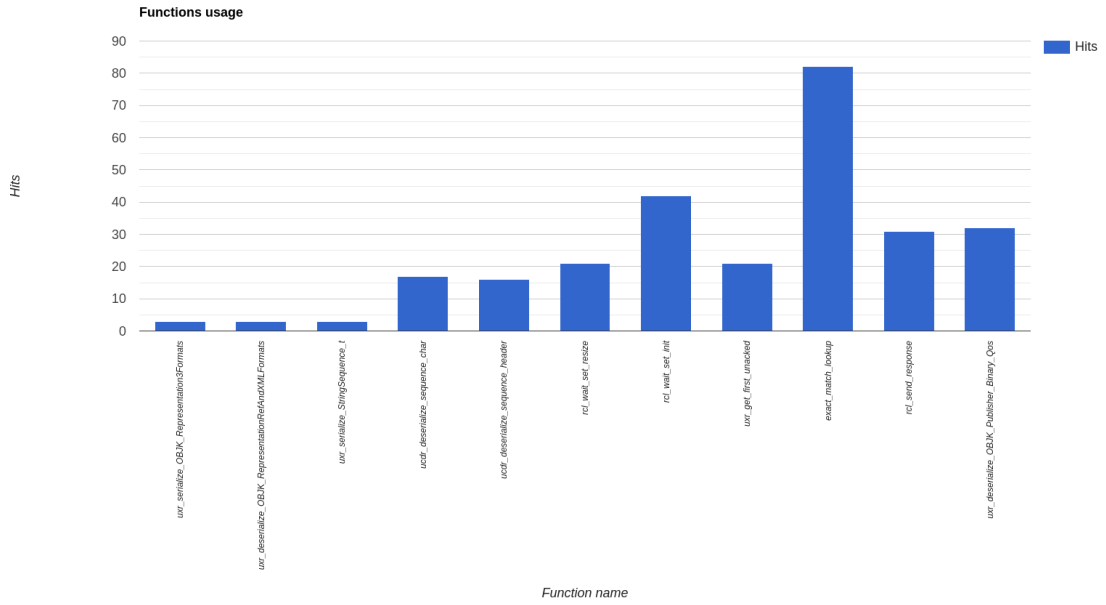
- [3]/6lowpan_publisher/sched/20210821_13_11_1629544300_sched.json [3],
- [3]/6lowpan_publisher/sched/hreadable.data [3].

The data showed that the Galactic version as similar execution performances as the Dashing version. Indeed, the application mostly waiting for I/Os to complete.

3.3.4 Functions usage

Here the functions usage is presented. Complete results are available here:

- [3]/6lowpan_publisher/fusage/20210821_12_29_1629541786_fusage_analysis.json [3]
- [3]/6lowpan_publisher/fusage/hreadable.data [3]



The functions that are called the most often (> 1 times) are:

- *exact_match_lookup* around 82 times,
- *rcl_wait_set_init* around 42 times.

The 6LoWPAN medium also shows that Galactic has a reduced number of different function called in contract to the Dashing version.

3.3.5 Static usage

At this point static memory usage is presented.

Complete results are available here:

- [3]/6lowpan_publisher/mem-static/20210821_12_06_1629540385_mem_static.json [3],
- [3]/6lowpan_publisher/mem-static/hreadable.data [3].

Below the static memory usage results:

```
{"microros": {"stack_used_bytes": 1720, "delim": 0}}
```

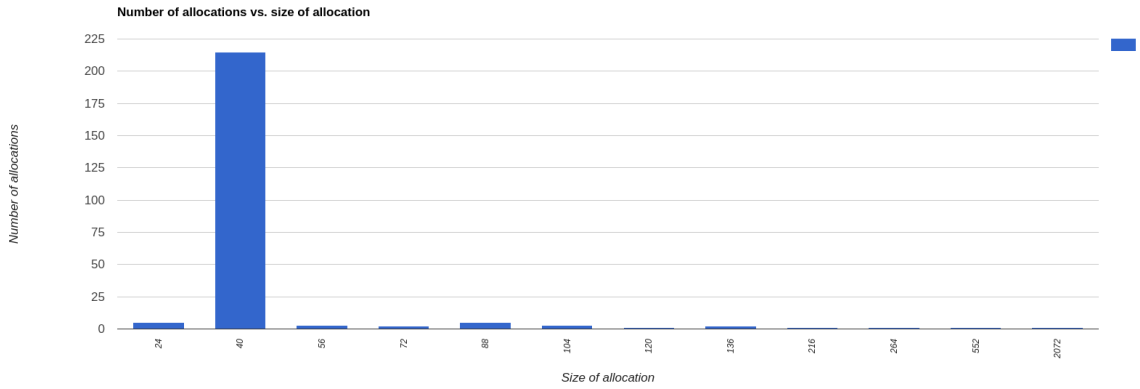
The stack usage in the Galactic version was reduced by a factor 8 in comparison to the Dashing version.

3.3.6 Heap allocation resources

Below are the results of dynamic memory usage measurements.

Complete results are available here:

- [3]/6lowpan_publisher/mem-dyn/20210821_12_53_1629543187_mem_dyn.json [3]
- [3]/6lowpan_publisher/mem-dyn/hreadable.data [3]



Regarding the heap allocation results, it's noticeable that:

The maximum size that is allocated twice by the publisher is 2072 bytes allocated by the 6LoWPAN stack.

The biggest number of allocations is 40 bytes 210 times.

The results from Galactic are very similar to Dashing.

3.3.7 Power usage

The results are available here:

- [3]/6lowpan_publisher/pwr/measure_info.txt [3],
- [3]/6lowpan_publisher/pwr/pwr_oscilloscope.PNG. [3]

According to the measurement made, the 6LoWPAN publisher is using around 740mW of energy. So in comparison, Galactic is slightly more efficient than Dashing.

3.3.8 Interrupts and interrupt service routines

As the same equipment was used with the same RTOS, the measurements regarding the ISR are leading to the same results. Therefore they would worthless.

3.3.9 Communication resilience

On the other hand many practical communication tests were performed during smart warehouse use-case preparation. The results of communication resilience are presented below.

Testing 6LoWPAN with 4 devices connected to micro-ROS Agent it was noticed that:

- the 6LoWPAN devices are sensitive to the presence of too many devices operating at the same frequency leading to connection drops and delays
- data integrity in the event of transmission disruptions is assured
- in case of single device, no significant transmission delays were found,
- two-way communications possible (publisher/subscriber).

3.4 Security assessment

There are no differences between the Dashing and Galactic versions.

Micro-ROS uses the resource-optimized DDS for Extremely Resource Constrained Environments (DDS-XRCE) standard, implemented by eProsima's Micro-XRCE-DDS. As was mentioned before there are no implemented authentication and encryption capabilities into Micro XRCE-DDS. This feature is difficult to implement in microcontrollers, as authentication and encryption are computationally expensive. On the other side, Fast DDS and its RTPS wire protocol, mediating the interaction between the agent and the DDS world do implement security. It is a part of the protocol external to the micro-ROS project. So, it is not planned to make any specific benchmarking regarding security in Fast DDS.

Despite these difficulties, a short analysis of micro-ROS security will be presented. There are many methodoligies to do security assesment. For this this chapter, methodology described in [4] was used.

3.4.1 SWOT

SWOT analysis is a tool that help to identify strengths, weakness, opportunities and threats. This type of analysis is mainly used in related to business competition or project planning [5], but can be also used in other fields.

3.4.1.1 Strengths

- DDS implementation
- maintained by open-source community
- assured messages integrity,
- fully configurable QoS (Quality od Service) settings.

3.4.1.2 Weaknesses

- no implemented authentication capabilities,
- no implemented encryption capabilities.

3.4.1.3 Opportunities

- due to open-source license security can be developed / maintained by community
- better communication quality than in the case of custom protocol,
- authentification and encryption can be imlemented on microcontrollers with built-in security features like hardware crypto module.

3.4.1.4 Threats

- easy fake data injection (especially with radio connection)
- possibility to sniff the packets,
- sensitivity to overload due to limited resources of microcontrollers.

3.4.2 Conclusions and recommendations

The issue of security in communication with microcontrollers is specific. For the direct cable connections (Ethernet and Serial) authentification and encryption is not as important as for wireless connections (6LoWPAN). Therefore, when using radio communication, special attention must be paid to the above-mentioned risks. Despite the lack of encryption, the transmission can be secured at higher software levels. For example, using FIWARE Context Broker token for the operation can be generated, so it is impossible to make operation without the supervisor's permission. If encryption is required, a more efficient processor can be used, for example STM32 with security features like hardware crypto module [6]. Because XRCE-DDS is based on DDS it takes all its advantages, including QoS mechanisms.

4 Conclusions

In general we can see that the application is performing good in comparison to Dashing:

- The amount of bandwidth used is lower
- The stack memory is reduced
- The number of functions called during execution is reduced,
- And as result the energy used to run was reduced by 10% in average

Also, those benchmarks target the application itself which means that the lower layer are not taken into consideration. For instance the application is running different medium that can take a certain amount of memory (drivers are dynamically allocating some buffers and other needed structures).

The choice of the medium will be depending on the performance and what is possible to achieve in the environement where the micro-ROS based device will be running in. For instance:

- For low power two solution are available: UART/6LoWPAN
- For performance: Ethernet,
- For EMC constrained environement: UART.

The obtained results show that micro-ROS can be successfully used at the present stage of development.

References

[1] M. M. Tomasz Kołcon Alexandre Malki, 'D5.4'. [Online]. Available: http://ofera.eu/storage/deliverables/M32/OFERA_55_D54_Micro-ROS_benchmarks_-_Final.pdf

[2] PIAP, 'D5.4 results'. [Online]. Available: https://github.com/micro-ROS/benchmarking-results/tree/master/aug2020

[3] PIAP, 'D5.7 results'. [Online]. Available: https://github.com/micro-ROS/benchmarking-results/tree/master/aug2021

[4] D. Baghdasarin, 'MRO cybersecurity swot', *International Journal of Aviation, Aeronautics, and Aerospace*, vol. 6, no. 1, p. 9, 2019.

[5] Wikipedia, 'SWOT analysis'. [Online]. Available: https://en.wikipedia.org/wiki/SWOT_analysis

[6] 'Introduction to stm32 microcontrollers security - an5156'. https://www.st.com/content/ccc/resource/technical/document/application_note/group1/9f/0b/e4/b6/75/15/4f/e2/DM00493651/files/DM00493651.pdf/jcr:content/translations/en.DM00493651.pdf.