Monitoring and controlling of modular meshed networks

* Atanas Tanev [1], David Timmermann[1], Timothee Buettner [1], Johannes Mangler [1], Arne Roennau [1] and Ruediger Dillmann[1]

[1]*FZI Research Center for Information Technology, Haid-und-Neu-Str 10-14*
76131 Karlsruhe, Germany
E-Mail: {tanev,timmermann,buettner,mangler,roennau,dillmann}@fzi.de

ABSTRACT

Modular satellites are the new upcoming trend in commercial and non-commercial spaceflight. Successful concepts have been presented adequately in the past, which leaves the question open what consequences will come with a modular satellite system. The distribution of energy and information within the satellite becomes a greater challenge since partial failures could lead to a total loss of all modules. This paper focuses on the robustness of modular satellites and proposes a system to monitor and control individual modules or a complete satellite. The presented GUI serves as additional system information for the operator and increases the systems redundancy. The method for detecting unknown module configurations as well as verifying known configurations will be explained as well as advantages and possible improvements to the monitoring system.

1 INTRODUCTION

The concept of building modular satellites has gained momentum in the past years. They offer many advantages compared to conventional, monolithic satellites as described in [1]. From a system engineer point of view, such satellites can be seen as modular meshed networks. With each module of the satellite representing one node and the loop free data connection between all models representing the network mesh. This mesh is not bound to one topology and can even change during the operational time of the satellite.

Such a system poses a challenge to current monitoring and controlling systems of satellites, which in most cases require a monolithic system architecture with a low degree of autonomy.

Single nodes or complete parts of the satellite might become unavailable temporarily or permanently due to external events or reconfiguration steps. Running processes might need to be transferred from one onboard computer (OBC) to another since other modules of the mesh might be temporarily or permanently unavailable. The need to actively distribute tasks has ever been present and is best documented in long-time missions: it is necessary to distribute and balance the energy consumption and temperature level of system components to avoid permanent failure. Satellites at the end of their lifespan require carefully planned power schedules and need to have their components switched on and off accordingly. Enabling a system without continuous connection to monitor and control the satellite is an important step in modular aerospace. The proposed monitoring system is hence not only interesting for modular satellite systems but also for complex monolithic systems.

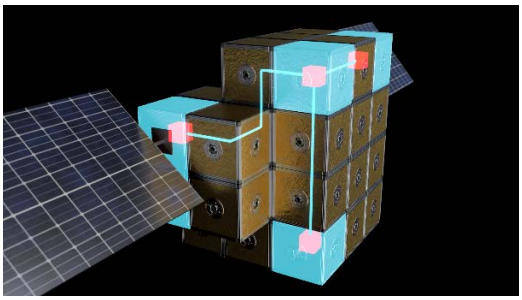

Figure 1: A possible configuration of an iBOSS satellite with a network mesh connection

We developed this system for modular satellites within the research project iBOSS (intelligent Building Blocks for On-Orbit Servicing and Assembly), granted by the German aerospace agency (DLR). In this paper, we propose a solution to the problem of monitoring and controlling a partly autonomous modular satellite with focus on distributed computation and active rescheduling of tasks. Our approach aims at establishing a unique interface between the satellite and a ground control station. We discuss advantages and disadvantages of our approach and evaluate the feasibility on a network simulator in a lab environment.

2 RELATED WORK

Modular networked networks can be found in various forms. While classical systems are data centers, modular robot systems are an abstract

version of such networks. An example of such a system is KAIRO 3[2]. The KAIRO 3 (Karlsruhe autonomous inspection robot) robot system is a segmented robot that was designed for rescuing and inspection tasks. It consists of multiple segments each composed of a drive module and a joint module. In contrast to other modular robotic systems like the PolyBot[3] each segment of KAIRO 3 has the ability to operate as an independent robot system or segments can be coupled to work as one robot. The concept of KAIRO 3 is comparable to that of a modular satellite. An example of such a modular satellite system is the iBOSS concept. A satellite consists of several similar modules called iBLOCKs. During runtime, these can be rearranged or additional iBLOCKs can be added.

Although monitoring and controlling of modular meshed networks has been researched in depth [4]. Solutions are difficult to apply for space applications. Future concepts of modular satellites will, however, depend on smart and reliable monitoring and controlling systems. Currently existing system for monitoring and controlling satellites, for example of cube satellites[5], are limited to one project or manufacture due to restricted software and hardware.

Through the usage of an open source software like ROS[6] (robot operating system) or its successor ROS2[7] controlling and monitoring systems are becoming independent of a specific setup and can become more versatile.

Like in earth based modular meshed networks it is necessary for the operator to get a clear understanding of the current connection layout of a satellite. For satellites like iBOSS with multiple connection interfaces on each building block the routing table does not offer enough information to determine the correct layout. For acquiring such a kind of information techniques like SNMP[8] can be used. SNMP-agents running on each module can request special information like the one hope neighbor of a specific port on a switch. Through this information and the knowledge of the structure of a module the complete layout of a satellite can be reconstructed during runtime.

3 APPROACH

In the context of iBOSS a modular meshed network is established through an on-board system in each building block which consist of an OBC (on board computer) and a switch. Furthermore each module is equipped with up to six 4-in-1 standardized interfaces[9] for mechanical coupling, electrical power, data and thermal interconnection. Through connection of multiple building blocks a satellite is build up and a loop free routing graph is established.

During runtime the system may experience different scenarios which lead to a new configuration of the satellite. These scenarios can occur naturally like the destruction of a module through space debris or hardware malfunction. Or might be initiated through the operator by initializing a reconfiguration process or by the addition of more building blocks. In all cases it is necessary to verify which module is still available in the current satellite configuration and how all building blocks are physically connected. After discovery of the new configuration it is necessary for the operator to check if in context of load and power distribution processes need to be transferred to another module. Modular satellites tend to poses a higher level of autonomy in contrast to normal satellites. Since decision of the satellite operating system might not be obvious on first glance the monitoring and controlling interface needs to give a clear reasoning for actions taken by the satellite. Therefore a highly flexible logging and notification system is needed which is capable of handling different levels of satellite complexity. Finally yet importantly, if an intervention of the operator is necessary the action has to be integrated into the control stream of the at least partially autonomous satellite.

While for all of these requirements solutions are present they are fitted to the corresponding project and therefore only useable for our purpose with great afford and drawbacks. Therefore, we see the need of a more generic monitor and controlling system for modular meshed networks. This approach is supported by the general software framework in iBOSS which is partly using ROS2. ROS2 is a framework with an integrated interprocess communication concept. Its predecessor ROS is used in many robotic applications. ROS2 already offers a wide range of tooling like broadcasting information through the network. Through using ROS2 functionalities it is possible to achieve a highly generic approach which is not limited to one specific system or project.

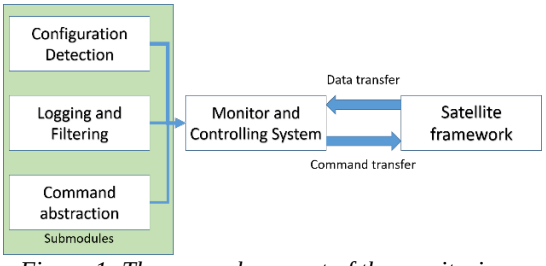

Figure 1: The general concept of the monitoring and controlling approach

A universal monitoring and controlling interface of modular meshed networks therefore has to provide three main features. First the logging of actions undertaken by the operator or the satellite operating system as well as status information. The capability to analyze and display the current configuration layout of all modules. As well as the integration of user action into the general control flow of the system. In the remainder of this paper approaches for monitoring and controlling are presented and evaluated:

Logging:

Status messages already generated in the software framework of the satellite are evaluated and stored in the corresponding category so they can be made available to the operator. A filtering mechanism is added to enable later comprehensibility of actions undertaken by the system.

Satellite configuration detection:

An existing layout file of the satellite is continuously evaluated against the status information of each module. All currently active modules are displayed and how they are interconnected.

Operator action integration:

Actions executed by the operator are evaluated if they can be transferred directly to the satellite or need to be splitted into sub actions and then being transmitted in the correct order to the satellite system.

Figure 2 presents the concept of the monitoring and controlling system.

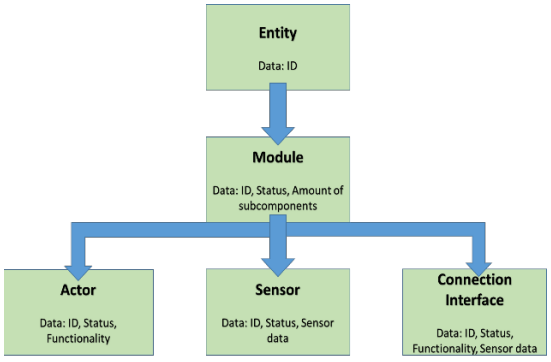

Figure 3: The general concept of the monitoring and controlling approach

4 Implementation and Tests

All implementation has been done and tested on the evaluation boards of the OBC within the iBOSS project. For other hardware like switches functional comparable hardware has been chosen due to availability of the real components.

3.1 Logging

For logging we make use of functionalities already being provided by ROS2 because parts of the satellite software framework are based on ROS2. The functions were already present in the system so there was no reason for reimplementing a different logging approach. Though a basic logging capability was given by the software framework we implemented extension to for the monitoring and controlling system.

As proposed before the idea of the system is to be easy transferrable to other modular meshed networks. Therefore we implemented a logging mechanism which is capable to monitor very detailed components in a system as well as basic components. The only requirement of a system to work with the logging mechanism is implementing an ENTITY. This is a basic upper class and only requires a unique ID to be identifiable in the monitored mesh. All other components that should be monitored by the system have to inherit ENITITY to acquire the ability to be monitored. In the context of iBOSS we took the top down approach. This approach is displayed in Figure 4.

All data are directly monitored on the satellite itself and stored in a corresponding database. All logged data is categorized into different criticality levels and assigned to the corresponding component to allow a specific view on selected components. The available data can be requested by the operator through the monitoring and controlling system and is made available through a suitable connection between the modular meshed network and the operator. The request for data can be made cyclical and the refresh time can be parameterized. This in combination with a filtering of components that shall be monitored can lead to a significant reduction of load need for allowing the operator to monitor the system. A screenshot of the logging mechanism can be seen in Figure 4.

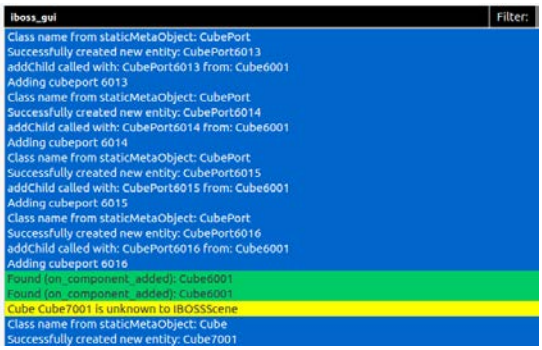

Figure 4: Screenshot of the logging mechanism

3.2 Satellite configuration detection

For detecting the current satellite configuration we first analyzed the potential of SNMP. As a precondition each switch and OBC present in the system has to implement the SNMP functionality. Starting from scratch each node in the mesh would execute a SNMP agent and a SNMP manager. A manager can request information from other agents. One of the information is the IP-/MAC-address of a module on a designated connection interface. Using the IP-address a directly connected module can be addressed to get information about the module for example a unique id. In combination with the knowledge of each building block and where a connection interface is physically located the current configuration of the mesh can be detected.
At first glance this approach seemed best suited for generating the current mesh configuration. After an expanded search we decided against the use of SNMP since the amount of available systems which implement the full feature set of SNMP, especially for space applications, is limited. Since our approach is meant to be as generic as possible we developed an alternative approach which does not depend on SNMP.
Our implemented approach uses a XML-File that stores the planned configuration of the satellite. That such a file exist is plausible since the purpose and structure of a satellite has been planned before. The XML is build up like this:

Unique ID of first module
    ID of module on left connector
    ID of module on the right connector
Unique ID of second module
    ID …

For detecting the current mesh configuration status data also used by the logging system is analyzed. The ID of all cubes that currently publish status data is recognized and compared against IDs in the XML-File. Matched IDs are enabled in the configuration structure and if two cubes which are marked as directly connected in the file are recognized a connection is made in the configuration view. In Figure 5 a detected connection is presented.
The result is a representation of all currently active modules. If a module is not active because it has been damaged by space debris, for example, this and the corresponding connections are not displayed and the operator is informed that there is a corresponding discrepancy between the configuration plan and the real configuration.
The operator is also informed if the system is recognizing a cube which is not provided in the configuration plan. A corresponding message is displayed and the cube is displayed without a concrete position in the configuration.

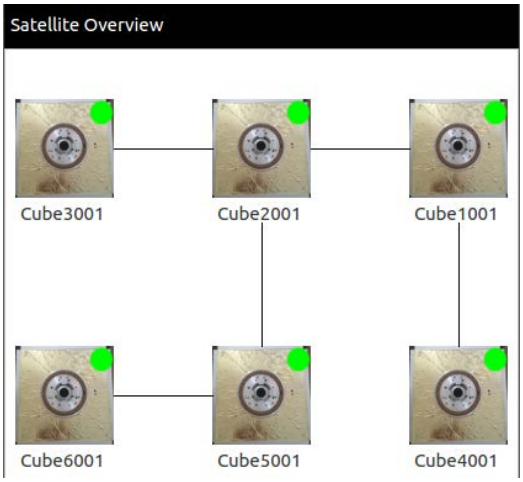

Figure 5: Detected configuration of the satellite

At any time it is possible to update the XML file in order to react to configurations that change at runtime. Otherwise a correct evaluation of the configuration of the satellite would only be possible during initialization.
The usage of only the unique IDs as identifiers is deliberately to allow easy transferability to other systems. This is also necessary because the building blocks in the iBOSS project are only to a certain extent identical and differ in the number of interfaces and payload.

3.3 Operator action integration

Since the software framework of OBC is based on Linux we have the possibility to use system methods from the operating system. While initially planned usage only for the framework to offer scheduling and distributed computing for load balancing and reliability. The same functionality can be used to integrate the control input of the operator directly into the control flow of the satellite. We use SystemD as a tool to handle the starting, preemption and restarting of processes on the satellite through the operator. The software framework of the satellites integrates these operations into the control flow of the system.
We use the daemon SystemD to start, stop or restart the process on the satellite. For controlling, the operator has the option of controlling processes directly or calling up a more general control function. The former is particularly useful if the process has no effect on other processes, for example starting a sensor evaluation. In the case of more complex control steps, such as the separation of a part of the mesh, i.e. the separation of a part satellite, or the load balancing on all modules the second option is selected. The monitoring and controlling systems determines the necessary control commands and transmits the desired actions to the software framework of the satellite which then reacts accordingly.
The controlling system also contains a feedback

function. If a control input could not be executed by the satellite framework the system informs the operator with a detailed notification. An example notification is shown in Figure 6.

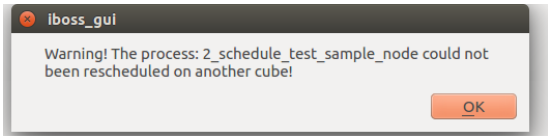

Figure 6:Operator notification after failed attempted to reschedule processes for load balancing

If the error can be traced back to a specific module. The corresponding module will be highlighted in the satellite component overview. A highlighted cube because of an error is shown in Figure: 7

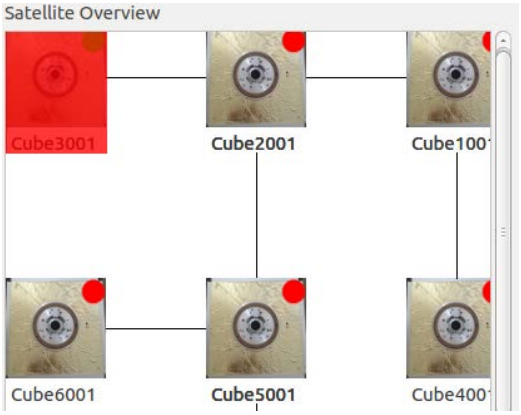

Figure 7:Operator notification of the position of a building block during a controlling error

3.4 Tests

The developed monitoring and controlling system has been tested in our lab on the hardware mentioned in the being of this section. All screenshots present in this paper have been taken from our tests.

The system showed high potential for effectively handling larger satellites with up to 20 building blocks. Tests proved that the system is able to operate on x86 as well as ARM processor architectures. This leads to a high flexibility regarding the control system.

Future tests will focus on original building blocks and not just on the correct OBCs.

4 CONCLUSION

This paper has presented a comprehensive tool for monitoring and controlling of modular satellite systems. The aforementioned implementation requires minimum processing power and is easily transferred to other similar systems. ROS2 was used and extended to monitor tasks over the entire network. It is capable of recognizing changes in configuration during runtime and is capable of accepting additional modules if new building blocks are connected. A downside of the system is the fact that an initial configuration needs to be given in the beginning. The capability to self-assess the satellite configuration is one of the future works in this project along with preparations for In-Orbit Demonstrations and further hardware test. The goal is to improve robustness for DSA time delay events and approve the system to run safely under space conditions.

Acknowledgement

This work has been co-funded by the German Aerospace Center (DLR) under national registration no. 50 RA 1503.

References

[1] T. Schervan et all., 2017, “iBOSS Modular Plug&Play – Standardized Building Block Solutions for Future Space Systems Enhancing Capabilities and Flexibility, Design, Architecture and Operations”, IAC 2017, Adelaide

[2] Pfotzer, Lars, et al. "KAIRO 3: A modular reconfigurable robot for search and rescue field missions." Robotics and Biomimetics (ROBIO), 2014 IEEE International Conference on. IEEE, 2014.

[3] Yim, Mark, David G. Duff, and Kimon D. Roufas. "PolyBot: a modular reconfigurable robot." Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on. Vol. 1. IEEE, 2000.

[4] Zilberman, Noa, et al. "Reconfigurable network systems and software-defined networking." Proceedings of the IEEE 103.7 (2015): 1102-1124.

[5] Babuscia, Alessandra, et al. "Development of cooperative communication techniques for a network of small satellites and CubeSats in deep space: The SOLARA/SARA test case." Acta Astronautica 115 (2015): 349-355.

[6] Quigley, Morgan, et al. "ROS: an open-source Robot Operating System." ICRA workshop on open source software. Vol. 3. No. 3.2. 2009

[7] Maruyama, Yuya, Shinpei Kato, and Takuya Azumi. "Exploring the performance of ROS2." Proceedings of the 13th International Conference on Embedded Software. ACM, 2016.

[8] Nagaraja, M. G., Ranjana R. Chittal, and Kamod Kumar. "Study of network performance monitoring tools-SNMP." IJCSNS 7.7 (2007): 310-314.

[9] Kortmann M., et al (2015). Building Block-Based iBOSS Approach: Fully Modular Systems with Standard Interface to Enhance Future Satellites, IAC-15-D3.1.3