VIDANA: A fault tolerant approach for a distributed data management system in nano-satellites

Thomas Walter*, Alexander Hilgarth, Tobias Mikschl, Sergio Montenegro

Aerospace Information Technology, University of Würzburg, Würzburg, Germany
**(e-mail: thomas.walter@uni-wuerzburg.de)*

Abstract – Reliability is always a concern in satellite technology and there are different strategies and methods in use to achieve this goal. The most typical attempt to achieve reliability today is to add an identical redundant device as a backup device in case of a failure. Due to the limitations in size and the constraints for mass or energy consumption it is not always possible to follow that strategy in the field of small satellites. In comparison among all involved subsystems of a space vehicle the data management system can be defined as the most critical spot of the whole system. The successful accomplishment of a mission will strongly rely on the reliability of the data management system. Unfortunately a fully redundant system cannot be implemented without adding size, weight and cost. This, however, is countered by the demand of more reliable systems along with decreasing budgets.

Index Terms – Spacecraft autonomy, data handling systems, fault tolerance, robustness, adaptive systems, distributed control, recovery, networking.

1. INTRODUCTION

Nano satellites are becoming more and more important for orbital and interplanetary missions. The requirements for such vehicles are steadily increasing. The missions demand autonomous operation, self-adaptability to different situations, self-healing of the system and many other self-x's. All of these objectives depend strongly on the capabilities of the data management system (DMS). VIDANA aims to provide a highly dependable and high performance computing systems based on limited physical resources and COTS (commercial off-the-shelf) components.

A VIDANA data management system is a network of software and hardware components. Our system approach is based on the middleware of RODOS (Real Time On board Dependable Operating System). This reliable interconnection network (software and hardware) can interconnect many unreliable redundant components such as sensors, actuators, communication devices, computers, and storage elements. Our goals are:

1. Dynamic adaptability, dynamic scalability for dependability and for speed.
2. Static adaptability (tailoring) for different missions.
3. Unified communication protocols for software and hardware.
4. Using the identical protocols for inter- and intra-spacecraft communication.

2. SOFWARE-IMPLEMENTATION

Our strategy to implement complex software (control) systems is to decompose the system into simple communicating building blocks (BB). The software BBs are similar to integrated circuits: only the interfaces (pin-out) and the behaviour have to be known to interconnect them. They can be seen as ports (like hardware-pins) on which the BB expects or publishes services in terms of messages (see Fig. 1).

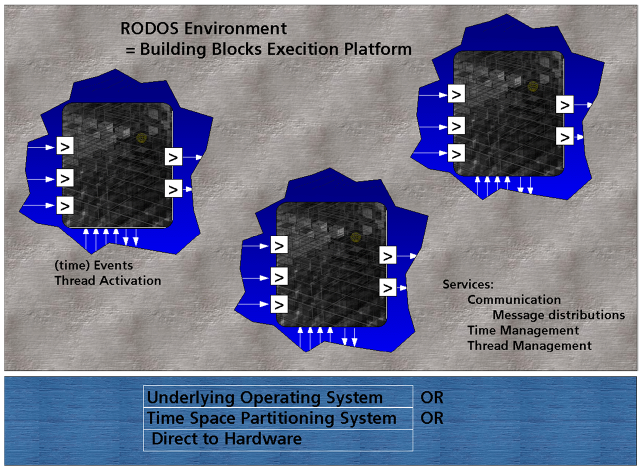

Figure 1: Software components / Building blocks

RODOS is an open source building block execution platform/environment designed for space applications and for applications demanding high dependability. The operating system is a fully pre-emptive microkernel system and it is implemented in C++ using an object-oriented framework. On top of this kernel the RODOS middleware distributes messages locally and globally (using gateways).

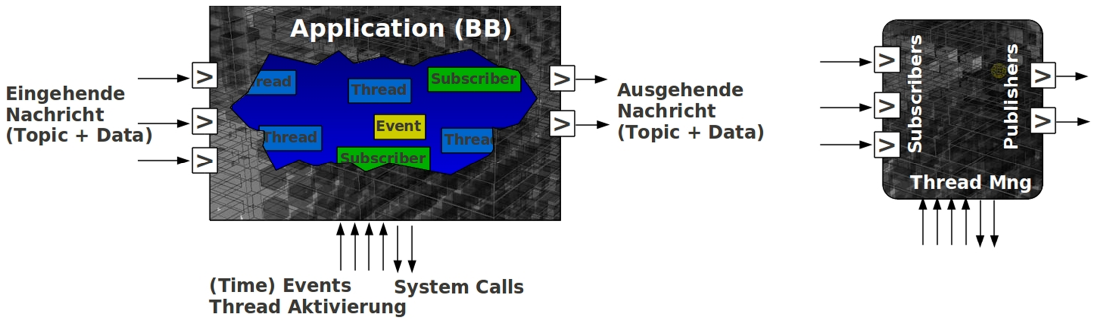

Figure 2: Interconnecting Building blocks

The RODOS execution platform provides a (software) interconnection network between applications/building blocks (the middleware). A building block requires some services (incoming messages) in order to be able to provide other services (outgoing messages). The execution platform distributes such services (messages) from producers to consumers. All communication in the system is based on a publisher/subscriber protocol.

A key functionality of a VIDANA system is the task migration. The idea is to detect a node failure and then to autonomously migrate all running tasks from the defect node into a healthy node. This is achieved by a software component called *task distributor* (TD), that is periodically scanning the complete system status for node health and task distribution. It will assign the tasks of the failed node based on certain rules. One rule is to keep the CPU load of all nodes balanced.

For critical tasks, this migration process might take too long (depending on the hardware up to several hundred milliseconds). In that case, the publisher/subscriber method of the middleware offers a great benefit: there can be several instances (replicas) of the same tasks on different nodes, publishing their outputs on the same topics. A *voter* then selects the best results for further processing. On the basis of the methods described above, a total system breakdown due to a single node failure appears extremely unlikely.

3. HARDWARE-IMPLEMENTATION

A VIDANA computer is a network of tiny computing nodes. Each node has an intelligent interface to the network, which is able to understand and execute the software protocols in the system. Attached to the node core we have some specialisations like flash mass memories, number crunching accelerators (DSP) and special IO devices to implement *intelligent* sensors and actuators. Nodes attached to the network can be turned off and on again at any time to be able to adapt to the current mission situation. The required tasks for the current mission phase will then be distributed among the active nodes. Tasks are able to flow from any node to any other. In VIDANA these tasks are denoted as *flowing tasks*. For different missions there can be different configurations of nodes. We consider a one-size-fits-all approach is not a feasible solution for all missions.

3.1 Overview of the different sorts of nodes used in VIDANA:

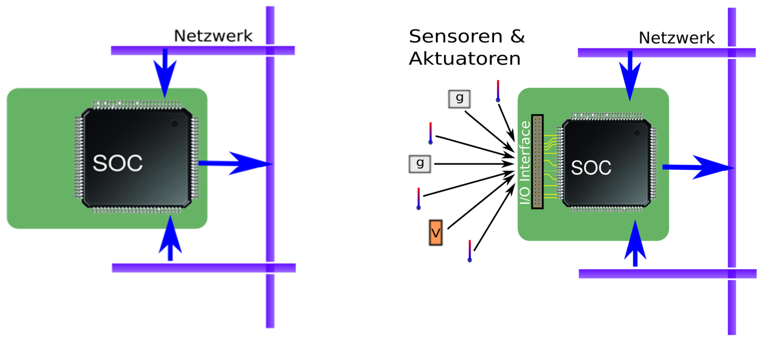

Figure 3: Basic node (left) and sensor/actuator node (right)

The ***basic node*** (Fig. 3 left) is implemented in a single SOC (System-on-a-chip) chip. This SOC implements all required interface to the network and implements (in software) the required communication and routing protocols. With the rest of the computing power (a lot) it can be used as simple computing node.

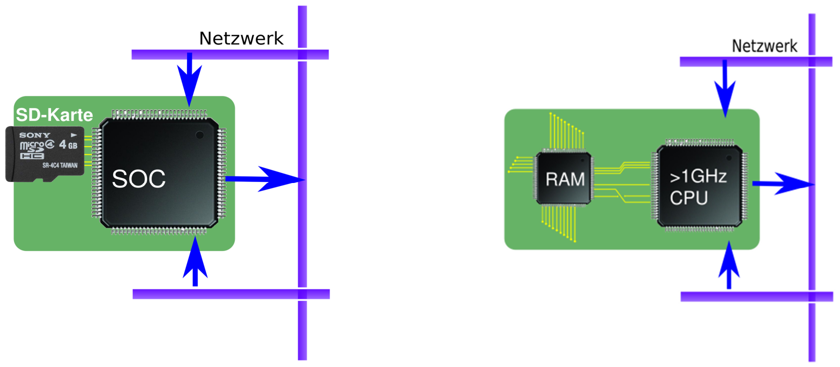

Figure 4: Mass-memory node (left) and number-crunching node (right)

As an extension of the basic nodes we may have different kinds of specialized nodes. The ***intelligent sensors and actuators node*** (Fig. 3 right) are similar to remote terminal controllers and transform the signals and protocols of (not intelligent) devices into the VIDANA network protocol. So it is possible to attach any intelligent device to the network and use it in a plug and play manner. The ***mass-memory-node*** (Fig. 4 left) is another extension of the basic node. It just adds SD-Flash-Memory cards and implements a file system. ***Number-crunching nodes*** (Fig. 4 right) are required to process a larger amount of data as is usual in the area of image processing/recognition, optical navigation, etc.

Each demonstrator node has three UART connections, so it can connect to three neighbour nodes. To allow an easy testing of different network structures all three connections of each node are connected to an FPGA. This FPGA is then configured to route the connections between the nodes in a certain pattern. In the illustrated mode (Fig. 5) only one UART connection to each node is used. The FPGA connects the transmit output of the node that is determined for sending to the receiving inputs of all other nodes. The acknowledge line of the sending node is set to high by the FPGA to indicate that the node is allowed to send.

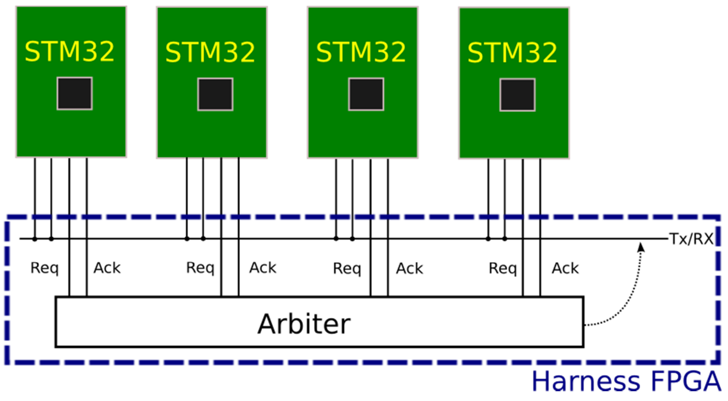

Figure 5: Bus-style network

In this mode only one node can transmit at a time and transmissions always reach every node in the network.