VIDANA: data management system for nanosatellites

Sergio Montenegro
Aerospace Informatics
University Wuerzburg
Am Hubland; 97074 Würzburg; Germany
sergio.montenegro@uni-wuerzburg.de

Thomas Walter
Aerospace Informatics
University Wuerzburg
Am Hubland; 97074 Würzburg; Germany
thomas.walter@uni-wuerzburg.de

Funded by the Space Agency of the German Aerospace Center (DLR) with federal funds of the German Federal Ministry of Economics and Technology (BMWi) under 50RM1203

Abstract: A Vidana data management system is a network of software and hardware components. This implies a software network, a hardware network and a smooth connection between both of them. Our strategy is based on our innovative middleware. A reliable interconnection network (SW & HW) which can interconnect many unreliable redundant components such as sensors, actuators, communication devices, computers, and storage elements,... and software components! Component failures are detected, the affected device is disabled and its function is taken over by a redundant component. Our middleware doesn't connect only software, but also devices and software together. Software and hardware communicate with each other without having to distinguish which functions are in software and which are implemented in hardware. Components may be turned on and off at any time, and the whole system will autonomously adapt to its new configuration in order to continue fulfilling its task. In VIDANA we aim dynamic adaptability (run tine), static adaptability (tailoring), and unified HW/SW communication protocols. For many of these aspects we use "learn from the nature" where we can find astonishing reference implementations

Funded by the Space Agency of the German Aerospace Center (DLR) with federal funds of the German Federal Ministry of Economics and Technology (BMWi) under 50RM1203

1. INTRODUCTION

Nano satellites and nano spacecraft are becoming more and more important for orbital and interplanetary missions. The expectations to such vehicles increases very fast. They shall be autonomous, self adaptable to different situations, provide self healing and many self-x's. Such vehicles shall be able to operate in swarm, cooperate for docking and auto assembly, and many other expectations. All this relays primary on the capabilities of the data management system (DMS). VIDANA aims to provide high dependable and high performance computing systems using very limited physical resources. We aim to exceed current data management capabilities using very simple (COTS) components.

Another challenge is the variety of different target missions. Developing a DMS which is applicable only for one missions would not be the best solution. A DMS which is good for every one ("one size fits all") would be too large in many cases and under-sized in other (almost never the right size). Our DMS should therefore be adapted with minimal effort to the specific mission (tailoring). At this point, let's differentiate between static adaptability which is performed by the developer for each mission (tailoring) and

dynamic adaptability which is performed autonomously by the spacecraft when circumstances change. Both are primary targets of VIDANA.

2. VIDANA TARGETS

Our targets are:

1. Dynamic scalability for dependability and for speed: At any time point we have a list of required tasks and redundancy and a list of available resources (computers, power, IO). The current resources will be dynamically allocated to tasks and redundancy to best fulfill the current mission requirements. The same resources can be used to achieve speed or to achieve dependability therefore when one increases the other decreases. Example: low speed high dependability: for critical operations like docking; High speed with lower dependability for example for image processing; low power (-> low speed, low dependability) for example for the cruise phases. The same mechanisms can be used to migrate software tasks among hardware components. This properties will be used to implement fault tolerance (self healing) and load balancing.

2. Static adaptability (tailoring) for different missions: highly (static) scalability without structural limits, down scalable to pico satellite boundaries. We take exemplar solution from nature for adaptability, communication strategies, distribution of services/capabilities and fault tolerance. VIDANA provides a building blocks architecture for both software an hardware and means to interconnect systems of different sizes. The systems is based on a service oriented architecture (for space applications)

3. Unified communications protocols for software and hardware. In this way services may be produces and consumed by hardware or software. For the produces (publishers) and consumer (subscribers) there is no difference if the communication partner is implemented in software or in hardware. This gives us a very flexible building blocks box which makes the tailoring tasks very simple.

4. The same communication protocols will be used inside (intra) and outside (inter) the space craft. This allow us to distribute services among different communicating spacecraft with no extra software effort. This is a basic support for cluster, swarms and assembly tasks. The same applies for the space segment <-> ground segment communication. This allows us to distribute tasks among space and ground and simplifies in an incredible way the commanding and telemetry tasks.

5. All these properties are focused on the requirements of nano and mini satellites, but can be used down to pico satellites and up to very big systems. Our software and hardware are absolutely ITAR-free.

3. IMPLEMENTATION

3.1. Software

Our strategy to implement complex software (control) systems is to decompose the system in simple communicating building blocks (**BB**). The whole application is then executed by a (software + hardware) network of simple BBs. Software BBs are similar to Hardware chips: Only the interfaces (pins) and their behavior has to be known to interconnect the components. In the same way we aim to define software components like in the figure 2, and 3. A software component provides input /output ports (comparable to hardware pins) which shall be interconnected by the run time system.

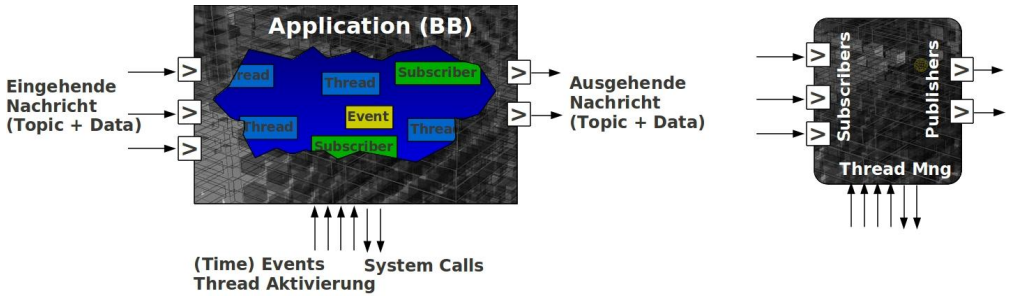

Figure 2: Software components / Building blocks

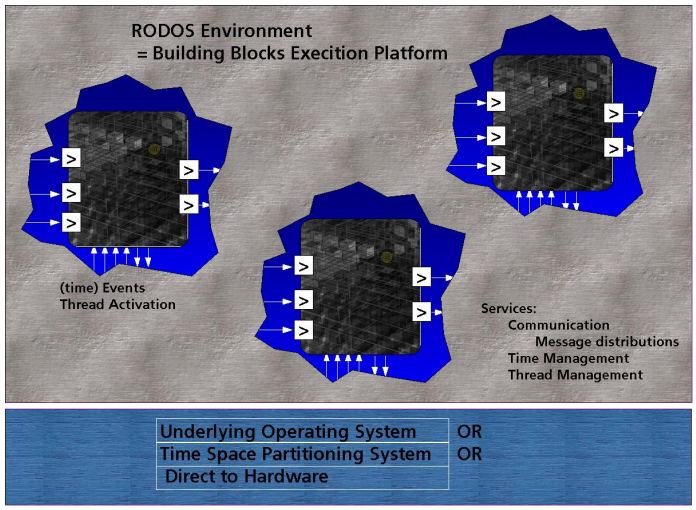

Figure 3: Interconnecting Building blocks

In order to be able to interconnect the building blocks (**BB**) we need an building blocks execution platform. We use RODOS (Real Time On board Dependable Operating System) which is an open source real time operating system and at the same time a building block execution platform/environment. RODOS was designed for space applications and for applications demanding high dependability. Simplicity is our main strategy for achieving dependability, as complexity is the cause of most development faults. The system was developed in C++, using an object-oriented framework simple enough to be understood and

applied in several application domains. Although targeting minimal complexity, no fundamental functionality is missing. Its micro-kernel provides support for resource management, thread synchronization and communication, input/output and interrupts management. The system is fully pre emptive and uses priority-based scheduling and round robin for same priority threads. This core functionality (kernel) may be executed directly on bare hardware (bare metal) or on the top of other operating systems.

On the top of this kernel the RODOS middlware distributes messages locally and using gateways globally. This middleware carries out transparent communications between applications and computing nodes. All communications in the system are based on the publisher/subscriber protocol. Publishers make messages public under a given topic. Subscribers (zero, one or more) to a given topic get all messages which are published under this topic. To establish a transfer path, both the publisher and the subscriber must share the same topic. A topic is represented by a topic ID and a data type. The middleware implements an array of topics which may be compared to hardware buses. Each time a message is published under a given topic, the middleware checks for all subscribers which wish to receive it. Each one will receive a copy of its content. To go beyond the limits of the computing node we use gateways which may read all topics and forward them to the network and vice versa (Figure 4)

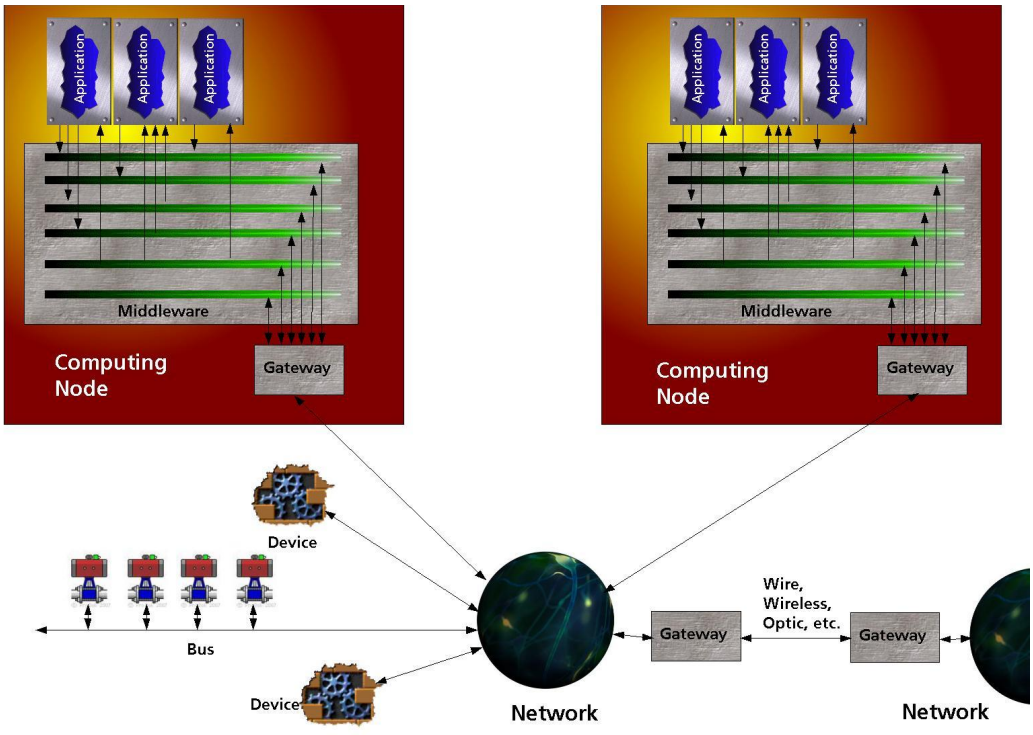

Figure 4: Global Communication using gate ways

3.2. Hardware

A VIDANA computer is a network of (very) small computer nodes. We concentrate our efforts on the communication system which includes software and hardware components. 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 communication interface we have a node core which is a micro controller with different IO interfaces, which will be not equally used in different nodes or missions. Attached to the node core we have some specializations like flash mass memory, number crunchers accelerators (DPS) and special IO devices to implement "intelligent" sensors and actuators.

Nodes attached to the network can be turned off and on at any time to be able to adapt to the current mission situation. The required tasks for the current mission phase will be then distributed among the active nodes. For fault tolerance purposes we shall have at least two active nodes at any time and important tasks shall be executed at least two times at any time point. Tasks are able to flow from any node to any other. We call this "flowing tasks"

For different missions we can have different configuration of nodes. we consider "One size fits all" is not a feasible solution for all missions. The only important aspect is the interoperability among different nodes so it shall be possible to attach any kind of node to the network and use it in a "plug and play" way without having to modify the software.