Resource sharing, communication and control for fractionated spacecraft (YETE)

T. Mikschl* S. Montenegro* A. Hilgarth* F. Kempf**

K. Schilling T. Tzschichholz*****

* Department of Computer Science VIII : Aerospace Information Technology,
University of Würzburg, Germany

**Department of Computer Science VII : Robotics and Telematics,
University of Würzburg, Germany

*** Zentrum für Telematik e.V., D-97218 Gerbrunn, Germany

Abstract

Modern spacecraft data handling systems are assembled of many highly specialized computer nodes, which are responsible for single subsystems. In this paper we want to present a more modular and distributed approach, in which computing power is shared among a network of computing nodes. Using wireless data links in this network of sensors, actuators and computing nodes enables flexibility and reliability regarding formations of various spacecraft. Furthermore wireless data connections enable not only intra-satellite, but also inter-satellite sharing of subsystems and computing power. As wireless links bring many constraints in regard of data link quality like delays and packet loss, we first simulate the distributed system with its various parameters.

1. Introduction

A system design decision for modern spacecraft (satellites, rovers, etc.) concerns the architecture of a robust on-board data handling (OBDH) and resource management in a mission. A common approach for OBDH is to use several specialized subsystem computers in parallel for the individual tasks, i.e. for sensor data post-processing and to hardwire the communication network of the individual subsystems of the spacecraft. These subsystems are then controlled by one or more redundant general purpose computing units (GPUs). One drawback of this centralized approach is that computing resources of the specialized subsystem computers cannot be shared among other subsystems or other spacecraft in a mission, which results in wasted computing resources. Furthermore should all GPUs fail, all still working spacecraft subsystems are lost for the mission.

2. YETE Overview

The fractionated spacecraft approach we present (YETE) addresses a distributed data processing concept with strong emphasis on modularity at hardware and software level.

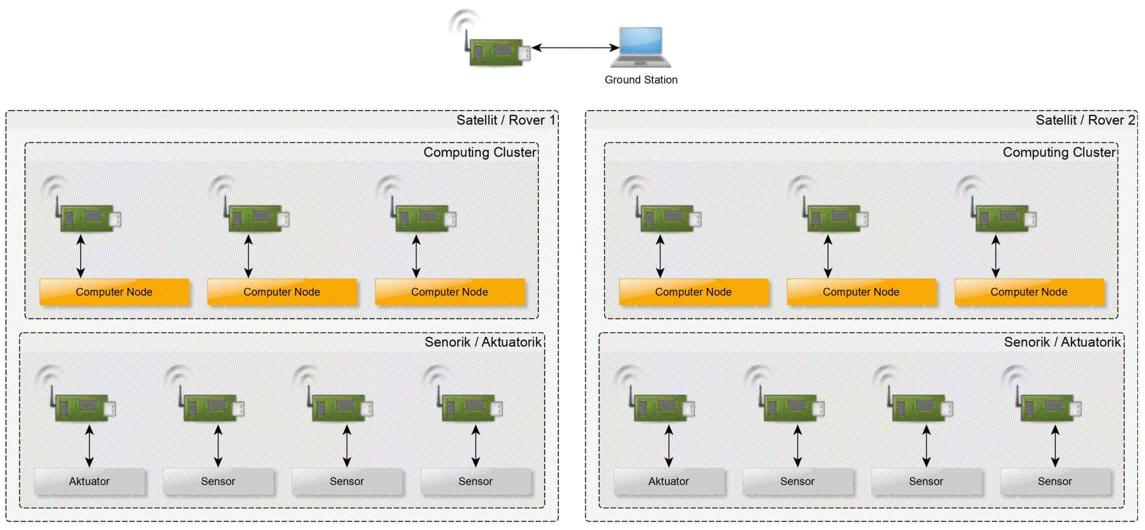

Figure 1: The YETE concept: Wireless links are used to connect a cluster of computing nodes to very simple sensors/ actuators. All sensor- or actuator-specific processing now takes place in one or more computing nodes.

On the hardware level all spacecraft subsystems (sensors, actuators, computing units, etc.) are treated as independent nodes, interconnected via low power short range wireless links. Most device specific computations are performed on general purpose computer nodes, which form a computing cluster. Additional long range wireless links allow the space vehicle to use the computing resources, sensors and actuators of other space vehicles and to share its own.

On the software level all functional software units, i.e. I/O drivers or applications, are encapsulated into independent "Building Blocks (BB)". They can easily be added or removed allowing fast re-configuration of the software system to changing mission conditions. Intra-/inter vehicle communication, task distribution and task execution is handled by the middleware OS RODOS [1] which runs natively on all nodes in the spacecraft.

To perform tests of inter-/intra- / spacecraft-to-Mission Control Center (MCC) communication and system behaviour under controlled transmission channel conditions in different spacecraft networks/constellations, we simulate all communication inside the Omnet++ simulation framework. This also speeds up the development cycle of our YETE hardware demonstration platform. The software link-interface we created between RODOS and Omnet++ is transparent to all BBs inside a node and can be exchanged by real communication hardware (Bluetooth Low Energy, Wifi, etc.) later on. Results for different routing algorithms (DSR, AODV), protocols (DTNs) and simulated RF-hardware are presented and discussed.

Distributed control of spacecraft in YETE is done in Simulink running as a RODOS task. The control performance under varying communication conditions (delays, bit-error-rate, etc.) and a unified (model and controller on the same node) vs. a separated control concept is discussed.

3. Communication Simulation

With YETE being a highly distributed concept, communication between the individual system nodes plays a major role in the overall system behavior. The transmission channel properties, such as delay, bit error rate or packet loss greatly influence the ability to perform distributed system control and the way in which tasks can be distributed or shared among nodes in the network. Therefore, to allow the testing of different control algorithms, task distribution concepts and RF link hardware under controlled conditions, a communication simulation concept for YETE was developed. We decided to use the discrete event simulator Omnet++ [2] as simulation environment.

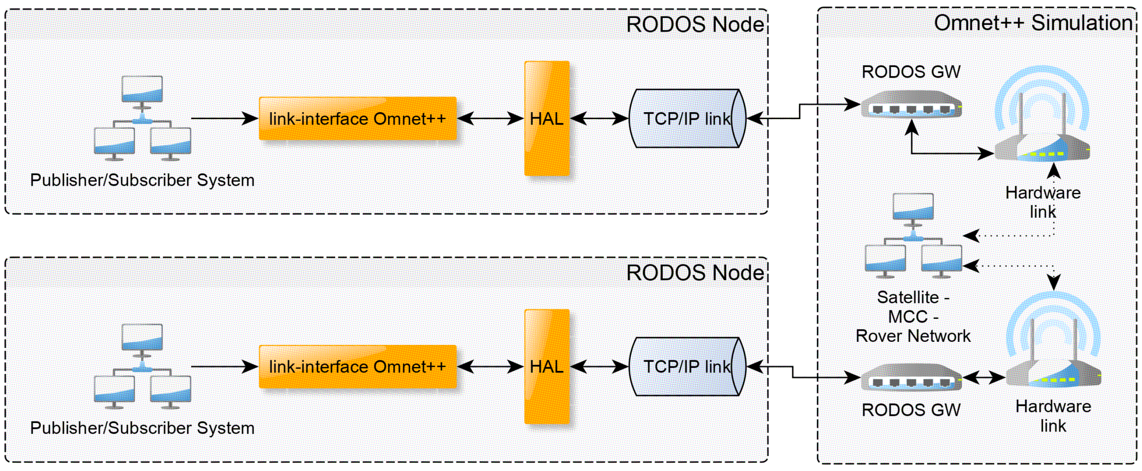

Figure 1: Integration of Omnet++ into the RODOS node communication layer.

Integration of external real-time applications into Omnet++ has already been done before [3], however none of the proposed concepts apply in this case. The reason is, that the external application, RODOS in this case, will be running on several distributed nodes, some of which are embedded devices. Therefore the link between the RODOS Operating System and the Omnet++ communication simulation consists of two parts, one is platform dependent and is running on the RODOS side and one is platform independent and is running on the Omnet++ side. The communication between the two parts is done via the TCP/IP protocol, preferably over an Ethernet cable connection. The part on the RODOS side is implemented as a RODOS hardware link-interface (omnetpp-linkinterface) over which RODOS topics can traverse. This link-interface can later be exchanged by link-interfaces of real hardware, like one of a Ultra Wide Band (UWB) or Bluetooth connection. This way the simulation appears transparent to the RODOS system and to the other software Building Blocks (BBs) running on the node. On the Omnet++ side a RODOS gateway module receives the topic data from one or more RODOS nodes via a two way tcp connection and forwards the topics to exactly one simulated hardware link inside the simulation environment. If the RODOS gateway module represents an intra-satellite link it also broadcasts a special satellite state topic to selected modules inside the satellite module which need the current satellite state, e.g. the satellite position or attitude, for their operation. One example for such a module is the satellite mobility module, which updates the satellite position inside the visualization map and which is also used in the signal strength calculations of the radio transmission links.

To facilitate realtime event processing and to handle new received RODOS data inside Omnet++ we extended the realtime socket scheduler of Omnet++, so that between two scheduled events, it performs a blocking select operation on the sockets of all RODOS gateways in the network and lets them handle newly received topic data if a socket event occurred.

4. Simulink Integration

For facilitating the development of control algorithms and high-level tasks, we developed a Simulink toolbox targeting RODOS applications. The toolbox consists of a support package (SP) which basically contains the glue code and interfaces to RODOS, and the block library, which provides specific interface blocks for connecting Simulink models with sensors or RODOS internals.

We have chosen Simulink, since this allows us to re-use existing models and develop new models very quickly. Also, this allows extending the models with other toolboxes and even interface to more hardware, when required.

Depending on the application, the controller requires measurements at a specific rate. The sensor responsible for these controller inputs must provide the samples at the required rate and send them to the controller via the RODOS middleware.

Using our support package, code can be directly generated from Simulink, such that a standalone binary is created containing a complete instance of RODOS and the model designed with Simulink. Later on, this binary can be copied into the flash memory of a microprocessor and can then be run on dedicated hardware.

The interface between the sensors, actuators and the model (i.e., the controller) is made by the RODOS middleware and its publisher/ subscriber architecture.

5. Demonstrator

To demonstrate the capabilities of the distributed system, we implement it on a simulation vehicle with sensors and actuators. Our goal was to replicate the systems of a satellite and its environment as close as possible. However we have to adhere to several constraints, like the limited financial resources and the space available.

The solution is a small vehicle, which uses on-board tanks with pressurized air and air pads to glide frictionless on a small air-film. On the top layer of the vehicle our computation system is integrated, consisting of the development boards combined with sensors and connected to the actuators.

The vehicle can be equipped with different sensors and actuators. For translation on a flat surface small air thrusters are used. Rotation can be handled by a reaction wheel in the center. Various sensor systems have been implemented on the hardware, for example a IMU with gyros and accelerometer, as well as a star tracking cameras which uses a printed star catalog on the floor as reference.

Acknowledgements

This project is funded by the DLR agency under grant no. 50RA1332, what is gracefully acknowledged.

References

[1] V. Petrovic S. Montenegro and G. Schoof. “Network Centric Systems for Space Application.” In: IEEE conference SPACOM. 2010.

[2] Christoph P Mayer and Thomas Gamer. “Integrating real world applications into OMNeT++.” In: Institute of Telematics, University of Karlsruhe, Karlsruhe, Germany, Tech. Rep. TM-2008-2 (2008).

[3] Andras Varga. “OMNeT++.” In: Modeling and Tools for Network Simulation. Springer, 2010, pp. 35–59.