HiPeRCAR: the High Performance Resilient Computer for Autonomous Robotics

P.A. Marra[(1)], D. Akuatse[(2)], E. Crudo[(3)], F. Fusco[(4)], S. Montenegro[(5)], R. Vitulli[(6)]

1	Pasquale Antonino Marra Alcatel Alenia Space Italia S.p.A.- Italy	marra.p@laben.it tel +39-02-25075-525
2	Daniel Akuatse Syderal S.A - Switzerland	daniel.akautse@syderal.ch tel +41-32-32338-9906
3	Emily Crudo Galileo Avionica S.p.A - Italy	emily.crudo@galileoavionica.it tel +39-02-3024-2843
4	Flavio Fusco Galileo Avionica S.p.A - Italy	flavio.fusco@galileoavionica.it tel +39-02-3024-2307
5	Sergio Montenegro - Fraunhofer Institut Rechnerarchitectur und Softwaretechnik (FIRST) - Germany	sergio.montenegro@first.fraunhofer.de tel +49-30-6392-1878
6	Raffaele Vitulli ESA ESTEC (TOS-ETD) – The Netherlands	raffaele.vitulli@esa.int tel +31-71-565-6791

Robotic applications require massive processing power not available in the today technology for Space: trajectory computations, 3-D translations, image elaborations overcome easily the crunching power of 300 MIPS. Moreover, the next exploration missions need to run without the support of human operators, requiring autonomous capabilities for re-planning their tasks in unexpected conditions. Thus, appeal to high-power computers is a mandatory requirement.

Where to find such a power? Can we still hope to perform complex robotic missions in the Space?

A generic high-performance payload controller, named "High Performance Resilient Computer for Autonomous Robotics" (HiPeRCAR), has been devised around a hybrid system where a reliable computer improves its processing capability by the help of more COTS-based computers. Rather than a distributed computer system, what is proposed is a co-processing system that guarantees reliability and power at expense of some acceptable operational discontinuities.

Leon2, PowerPC (but SPARC or DSP as well), SpaceWire network and Mass Memory are the main bricks of this system that the nowadays technology provides us. The system looks like a star of heterogeneous computer nodes around a switching router of SpaceWire links (Fig. 1). Most of the constituting equipments are today available or just ready to become available.

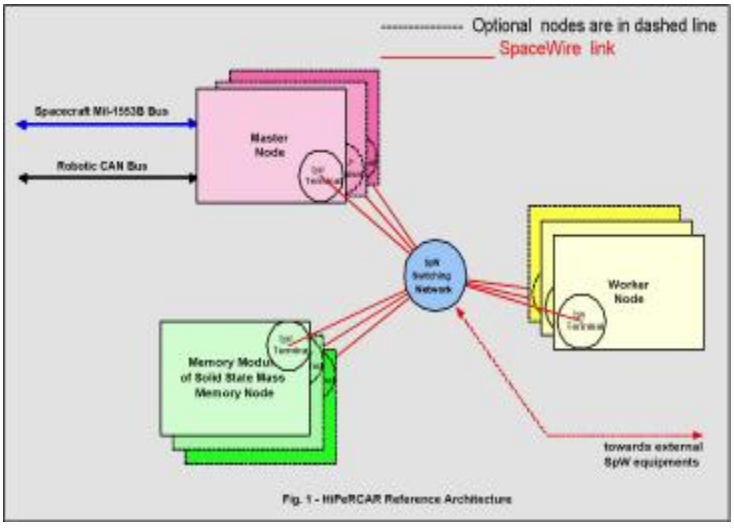

Fig. 1 - HiPeRCAR Reference Architecture

The HiPeRCAR design is inspired to the original MOSREM concept by ESA, which places radiation-hardened nodes as front-end of a pool of high-performance shear nodes based on industrial processors.
The front-end computer ensures system reliability and service continuity. The back-end pool of COTS nodes wants to provide non-dependable high-end computations.
A low-latency communication network and a data storing facility provide to the system the logistic support.

Thus, HiPeRCAR wants to combine the benefits of high performance COTS with the benefits of radiation-hardened Hardware to fulfil the requested requirements. The modular and scalable architecture allows also changes to the configuration without hardware changes.

The HiPeRCAR project aims at demonstrating the feasibility of such a dependable high-performance control system for space robotics, which guarantees continuity of service despite of failures and anomalies. It finds the wanted power in market-available PowerPC computer boards. The obvious weakness of those low-cost products is intentionally tolerated because solved by means of Software techniques. The resilience feature plays the most considerable role in the demonstrator, together with the management of the high-rate data transfer and the Solid-State Mass Memory facility.

For the purpose of demonstration a minimalist development approach is adopted with a minimum set of nodes and links; this configuration does not loose representativeness allowing to improve power, reliability and extension capabilities.
A Servo Control Unit (SCU) manages usually the low-level robot control loop. In the HiPeRCAR design, the SCU functionality is implemented within the Master node that throughout a CAN bus manages the sensor/actuator world.

A consortium of European companies lead by AASI (Laben department), is trying to demonstrate that it is possible to achieve this goal with the emerging Hardware technology and a sound Software technique. The team is working under the ESA contract ITT/1-4607/04/NL/AG that imposes some challenging programmatic constraints as:
- Use of European technology only
- Use of COTS boards
- Use of open-source Software components
- No license restrictions.

But also functional constraints as:
- High performance and low cost
- Fault tolerance and no replication of the resources
- Tolerance to crashes but not to the interruption of service

The first phase of the Project foresees to do the hardware and software design of a possible Space system. Two kinds of emulator will check over the second phase the features of this system:
- A Software simulator on commercial PC/Linux will run some study-cases to experiment and set-up the best strategy for managing a robotic Application spread over more computers.
- A physical target will be developed to allow a critical examination of the resilience capability by injecting faults into the system.

The test results will provide a leaned-lesson for the team, extended to ESA people, to base the strategy of the future robotic missions.

The paper describes the major features of HiPeRCAR Software and of the supporting platform.

The HiPeRCAR Middleware

The Middleware framework aims to offer the most simple and small possible interface to user tasks, which still provides all the required functionality and flexibility.
The Middleware includes real-time kernel, time management, resource management and communication functionality. Without an application the Middleware framework is inactive, it just reacts to interrupts doing nothing. An application adds actions to the framework by inheriting

classes and creating active objects. These objects will be integrated automatically in the framework. In this way the framework will be extended with User functionality.
In its widest extension, the Middleware framework includes also the hardware-dependent specific layer, which hides the physical characteristics for each node.

The same Middleware framework runs onboard all the nodes and carries out in uniform mode all the basic tasks taking care of a heterogeneous environment made of different computer types.
Its Communication Manager distributes messages among tasks everywhere located over the network. For the implementation there is no difference where the tasks will be deployed. The position of tasks can even change at run time, without requiring any explicit reaction of the other involved tasks. Task communicates using Middleware; its Communication Manager distributes messages among tasks everywhere located over the network but its only interface is messages. Tasks just define input and output messages; the rest is totally transparent (Fig. 2).

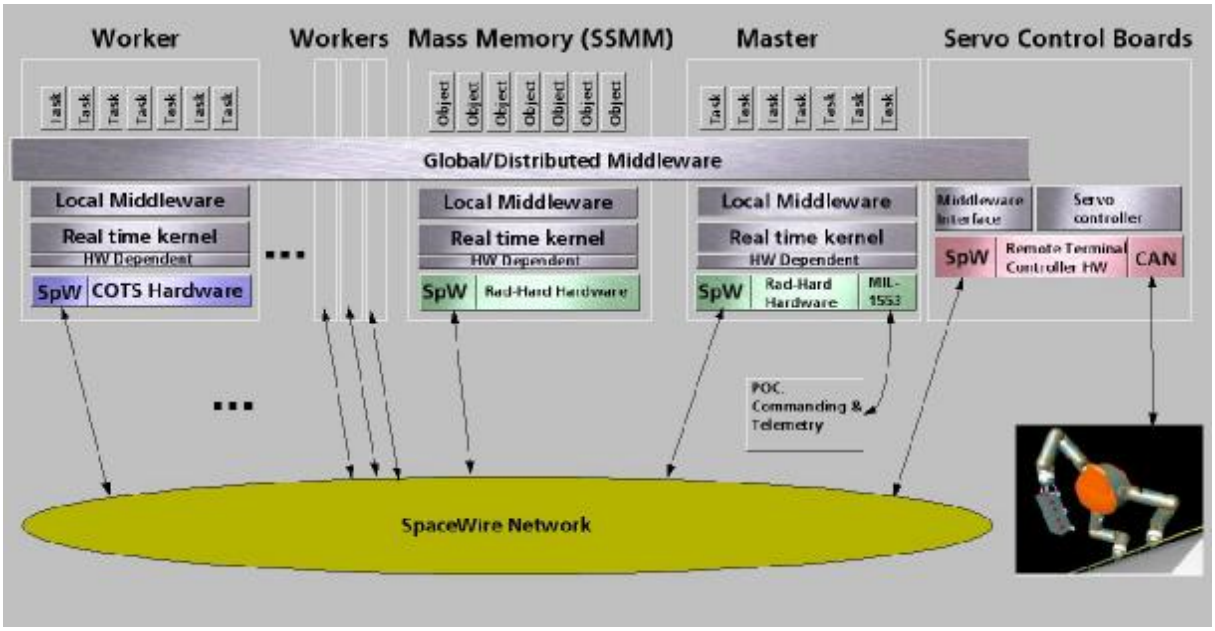

Fig. 2 - HiPeRCAR Software Framework

The robotic controller Application Software is implemented as a collection of building blocks on the top of Middleware, which creates a virtual interconnection network for all tasks running on it. The middleware allows the co-operation among the tasks over the system. It imposes no limitations to the communication paths, but the application can use/create a meaningful and efficient interconnected structure of tasks.

The HiPeRCAR system aims at the realisation of more robotic Missions in the Space; thus, there may be a specific configuration for each application. The system shall support to migrate from any of the required configuration to another only changing the upper application layer.

The application tasks run on top of a virtual global Middleware, which cross node boundaries and interconnect all threads/tasks in the system. They do not need to know the details of the lower layers as locations and hardware architectures.

The global middleware is based on a local middleware running on each node. This local middleware runs on the top of a real time kernel, which administrates time and local resources of the node. The real time kernel uses a virtual hardware abstraction, which is implemented within the hardware-dependent layer. This layer manages the peculiar features of the node: to move the Software system from a hardware platform to another one, only this small layer has to be re-implemented.

Fault Detection and Fault Recovery Strategy

HiPeRCAR has to provide continuity of service despite of failures in Worker nodes (the components not made of rad-hard technology). In case of failure, the shut down and reboot operations could last couple of seconds, still too much to give continuity to the interrupted service. Therefore a safe control shall relay on tasks running on Master node. The HiPeRCAR Software foresees that reduced versions (basic tasks) of the tasks running on the Worker Nodes are always active on the Master Node. Those basic tasks allow Master to control all the devices by alone (basic operative mode), but if advanced tasks are running on Workers (nominal operative mode), the results of such advanced services are used to improve movements and autonomy of the robotic system (Fig. 3).

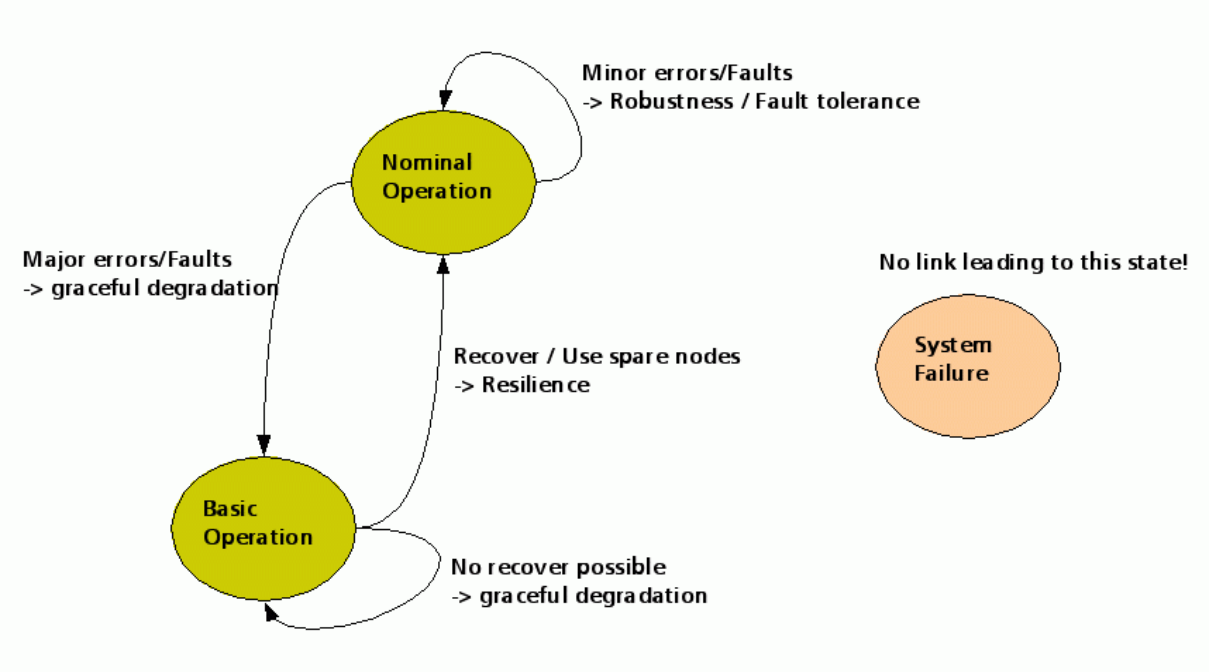

Fig. 3 - Operative Modes for FDIR

In this way, if a Worker node fails just the contribution of the advanced commands is missing, but the basic control task stays operable. From outside of the HiPeRCAR kernel, there is no discontinuity of service.

The System goes instantaneously and automatically from nominal to basic operations mode. In the meantime, the Recovery tries to reboot the faulty node. If this does not help, the faulty node will be turned off and a spare node will be activated if available. After the recovery task, a Configurator task reallocates the advanced tasks on running workers to go back to nominal operation (resilience). If this fails, then the system will stay in the basic operation mode (graceful degradation). In any case, there is no interruption of service.

Robotic Task based on Basic and Nominal modes

The splitting into Basic and Nominal modes for a given robotic function *F* is a major challenge. As there is no general rule to split a function in two complementary parts, this design activity has to be done on a case-by-case basis (Fig. 4).

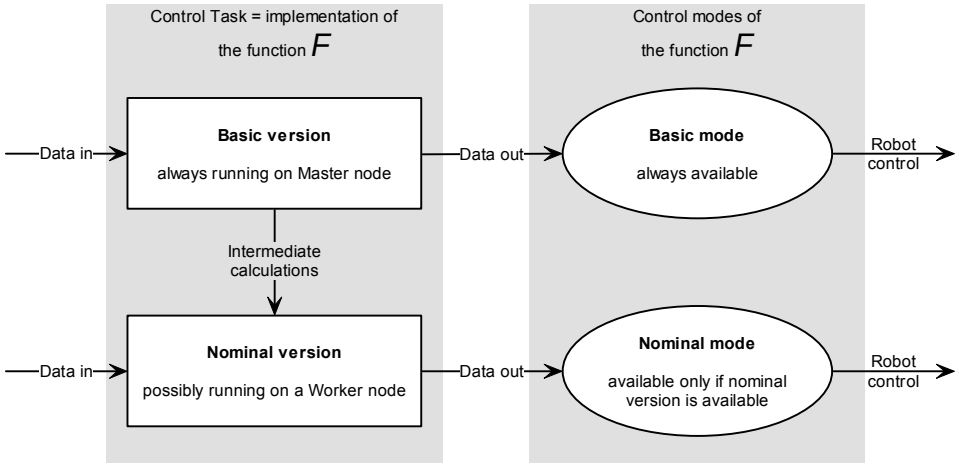

Fig. 4 - Basic and Nominal modes of a Control Task

The robotic scenario

In the HiPeRCAR framework, Middleware has fixed and mission-independent functionalities, while each set of mission-specific functions will constitute a particular system instance.
Two particular instances will be considered as reference scenarios on which the FDIR idea will be deployed: an orbital one, the Eurobot manipulator, and a planetary one, the Pasteur scientific payload on the ExoMars rover.

A simplified but representative version of the two systems will be considered and the mission-specific control tasks will be simulated, since the purpose is to verify the HiPeRCAR concepts (mainly the robustness/resilience mechanism). The tests will be conducted on DREAMS, a test-bed developed on the 3DRM visualization tool by ESA.

Eurobot mission scenario considers a robotic system composed of: three 7-joint robotic arms, three generic end-effectors, a centralized vision subsystem.
The Eurobot Control Tasks to simulate are: control of a single arm with end-effector; coordinated control of the arms, environment reconstruction, collision avoidance, mission autonomy, data acquisition.

Pasteur mission scenario considers a system composed of: drill, Sample Preparation and Distribution System, microscope.
The Pasteur Control Tasks to simulate are: drill control, SPDS and microscope control, mission autonomy, data acquisition.