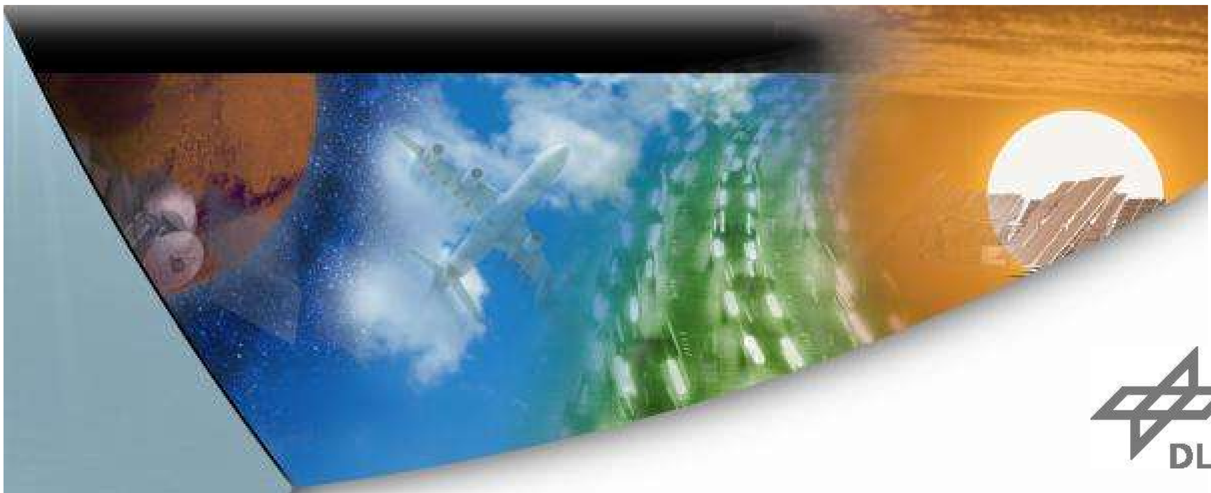


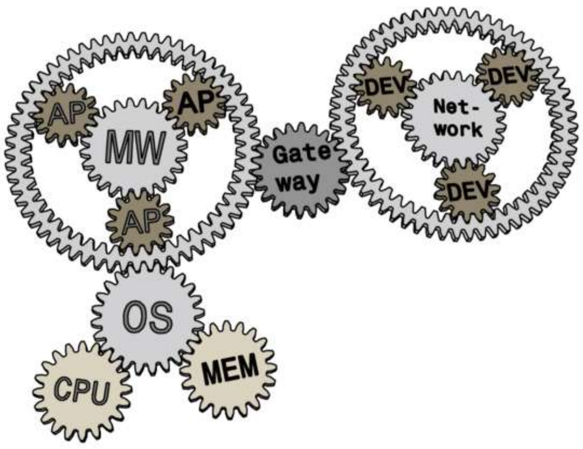
NetworkCentric Core Avionics

RODOS Introduction

Version:	**1.0**
Date:	**10.10.2008**
Author:	**Sergio Montenegro**

RODOS
Real time kernel design for dependability

The NetworkCentric core avionics machine consists of several harmonised components which work together to implement dependable computing in a simple way.



The NetworkCentric Machine

Computing units (CPU +MEM) are managed by the local real-time kernel operating system (OS) RODOS. On top of the kernel runs the software middleware (MW) of RODOS and around this middleware the user can implement its applications (AP). To communicate with external units, including devices and other computing units, each node provides a gateway to the network and around the network's several devices (IO Devs and computing nodes) may be attached to the system.

RODOS is a real-time embedded operating system (OS) designed 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, as its microkernel provides support for resource management, thread synchronisation and communication, input/output and interrupts management. The system is fully preemptive and uses priority-based scheduling and round robin for same priority threads.

RODOS provides a middleware which carries out transparent communications between applications and computing nodes. The messages exchange is asynchronous, using the publisher-subscriber protocol. Using this approach, no fixed communication paths are established and the system can be reconfigured easily at run-time. For instance, several replicas of the same software can run in different nodes and publish the result using the same topic, without knowing each other. A voter may subscribe to that topic and vote on the correct result. The core of the middleware distributes messages only locally, but using the integrated gateways to the

NetworkCentric network, messages can reach any node and application in the network. The communication in the whole system includes software applications, computing nodes and IO devices.

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. For this communication there is no difference in which node (computing unit or device) the publisher and subscribers are running. They may be in the same unit, or distributed around the network. They may be any combination of software tasks and hardware devices. To establish a transfer path, both the publisher and the subscriber must share the same topic. A Topic is a pair consisting of a data-type and an integer representing a topic identifier. Both the software middleware and network switch (called middleware switch), interpret the same publisher/subscriber protocol in the same way.

RODOS is the gearwheel of the NetworkCentric core avionics machine, which controls activities in the computing units/nodes (processors, CPUs, Memory).

The RODOS real-time kernel and middleware provide an integrated Object-oriented (OO) framework interface to multitasking resources management and to the NetworkCentric communication infrastructure. The RODOS framework seeks to offer the simplest and smallest possible interface to user applications, while still providing all the required functionality and flexibility. It includes time management, CPU and memory management.

The RODOS middleware provides communication between applications, networks and all devices attached to the network. The fault tolerance support implemented in the middleware allows us to create dependable systems using unreliable components. In our concept, a hardware failure is not an exception, but a normal case, which can be expected and has to be handled. RODOS redundancy management supports different strategies to provide the highest possible dependability, our target, which is the ultra high dependability using a principle which the world has since forgotten: *Simplicity.*

Simplicity does not mean, however, lack of functionality. Real time scheduling, resource management, synchronization, middleware and simple communication and all the functions one can expect from a microkernel are implemented – just as simply as possible. An important RODOS design target is the irreducible complexity; this is the minimal possible complexity for a determined function. When it becomes no longer possible to implement it simpler without destroying the functionality.

RODOS is based on very few and simple basic functions. Applications running on top of RODOS are implemented using object-oriented technology, resulting in highly modular application software. Applications running on the top of the RODOS middleware are built using the schema of software building blocks. Several (simple) building blocks (called applications) can be distributed and interconnected in a computer+devices network using the NetworkCentric core avionics protocols, to build more complex functionality. Building blocks can be implemented and tested independently of each other. Building blocks can be interchanged without having to modify other blocks or interfaces.