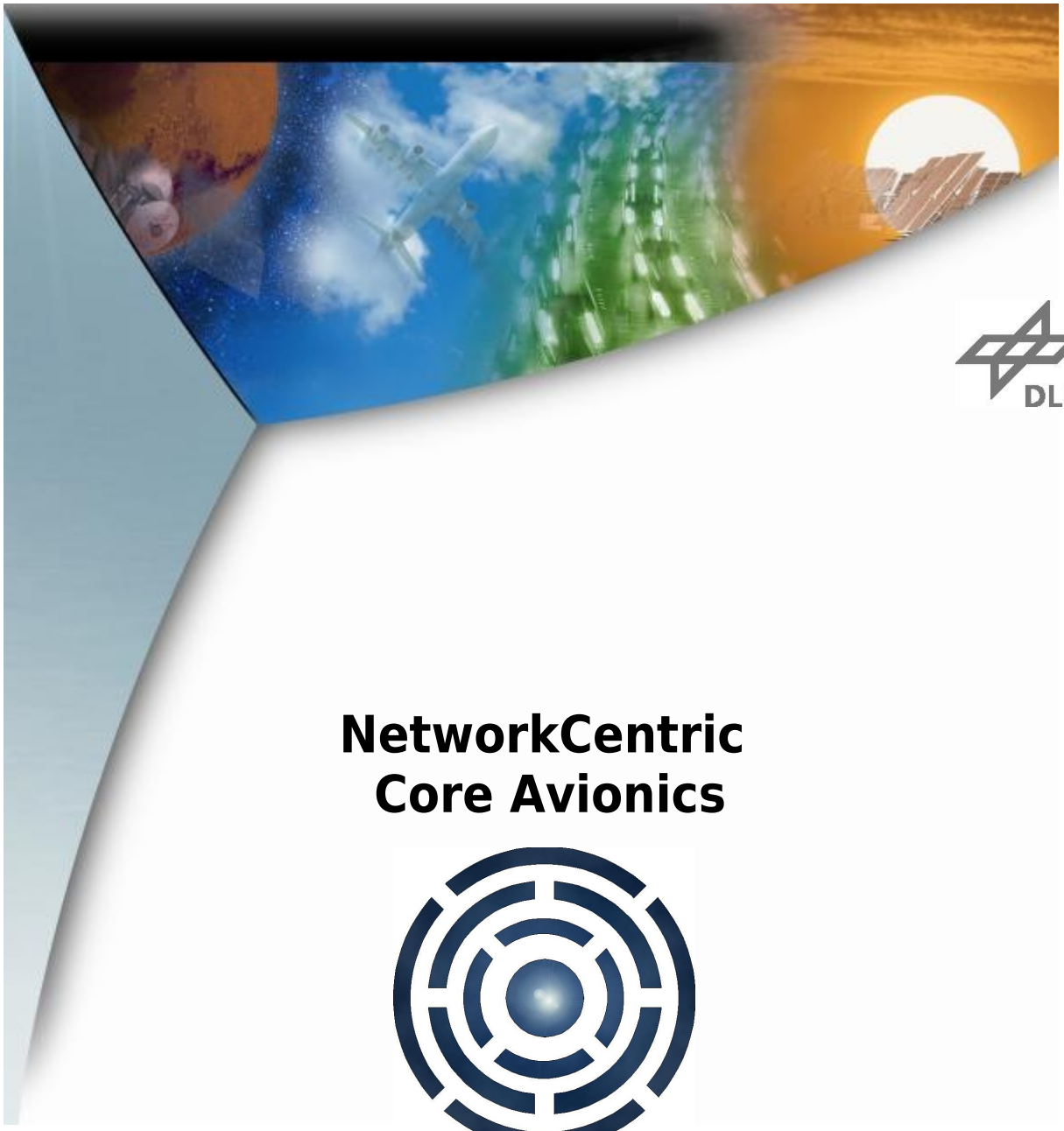

NetworkCentric Core Avionics

RODOS tutorial Middleware Distributed

Version: 1.0
Date: 10.10.2008
Author: Sergio Montenegro

RODOS Tutorial Middleware Distributed

From the middleware tutorial you have learned the procedures of how to communicate using the middleware. The communication described in that tutorial is only for local applications - all communicating units were running in the same node of the computer and in the same memory space.

The same programs – without modification – can be distributed to different computers and to different memory spaces. Each compilation only has to add a gateway to the network and for a network (simulator) to be started.

You may find two network simulators in the directory supportProgrammes. network and network_silent do the same, just that the network_silent has not printfs. network prints which nodes are active and which topics each node is expecting. In the same line it prints which message topics are being forwarded to each node. We recommend to use the first network. Use network_silent only if you have many nodes and many topics, when the prints are no longer readable.

How to use

Start several xterm windows.

In one window (at least 80x40 chars) start the network (or the network_silent) found in the directory support_programms.

You may try to read and understand the network implementation. It is not required for the tutorial as it is not as simple as the tutorial programs.

Keep the network running.

You may start with the already well-known application from tutorial_middleware.

See the **gateway.cc.** It just created an object called gateWayUdp.

You may copy it to tutorial_middleware.

Compile in one window all senders + demo_topic.cc + gateway.cc and start it.

In another window compile all receivers + demo_topic.cc + gateway.cc and start it.

The receivers shall get the messages from the senders, even if they are different Linux processes. At the network window you can view the message transfer activities.

You may start several senders and receivers in different windows. All of them shall communicate using the network simulator. Try to terminate some applications (using control-C) and after 2 or 3 seconds the network shall recognise the (simulated) “failure” and remove the corresponding node from the list.

you may start more than one network simulator at a time. One will work and the other will wait until the running network terminates (using control-C). Then another network will take control and the communicating nodes will continue their operation (fault tolerant).

2. Programmes

see mask.txt and support_programms/maskgenerator.

In this directory you will find:

a topic definition, for 6 very similar topics (demo_topic.h & demo_topic.cc)

6 senders (sender11.cc to sender 16.cc)

6 receivers (receiver11.cc to receiver16.cc)

mask.tst -> maks.h

All senders and receivers are very similar, they just use different topics, print on different positions on the screen and have different timing. The rest is practically the same.

There may be different gateways to different networks. But in this case we have only one gateway. It uses udp messages to communicate with the network simulator.

Sender11 corresponds to receiver11 and both use topicId 11.

It is the same for sender12 through 16 and the corresponding receivers and topics.

In every compilation you have to include the demo_topics.cc.

First try one sender and one receiver in the same window:

%linux-executable sender11.cc receiver11.cc demo_topics.cc

% tst

see how the receiver gets the messages from the sender and prints them,

now do the same in two different windows and a gateway (do not forget, the network shall be running in other window).

To execute it distributed on different windows (linux processes) please see the shell scrfipt %executeit

The receiver in Window B shall get the messages from the sender in window A.

Now try different senders and receivers in many different windows.

Combine different senders and receivers in the same window.

Stop and restart different windows.

Do not forget: After each compilation tst will be overwritten. Compile and execute for each combination.