ENABLING RAPID DEVELOPMENT OF ON-BOARD APPLICATIONS: **SECURING A SPACECRAFT MIDDLEWARE BY SEPARATION AND ISOLATION**

Andreas Lund, Carlos Gonzalez Cortes, Zain Haj Hammadeh, Fiona Brömer, Glen te Hofsté, Daniel Lüdtke

Institute for Software Technology, German Aerospace Center (DLR)

Motivation

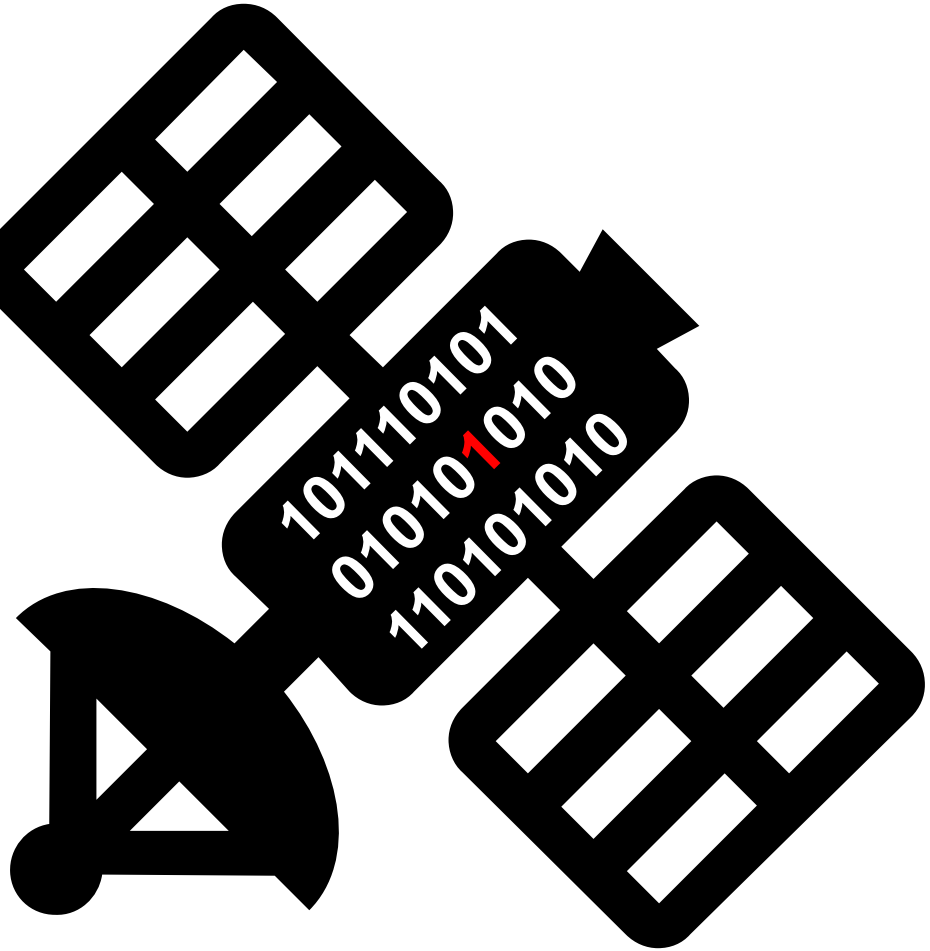
GOAL: Enabling rapid prototyping for applications on spacecraft

Let‘s test some software in space

- Space → harsh environment for electronics & software
  - radiation
  - few maintenance opportunities
- Avoiding loss of mission
  - critical

  → software quality needs to be assured

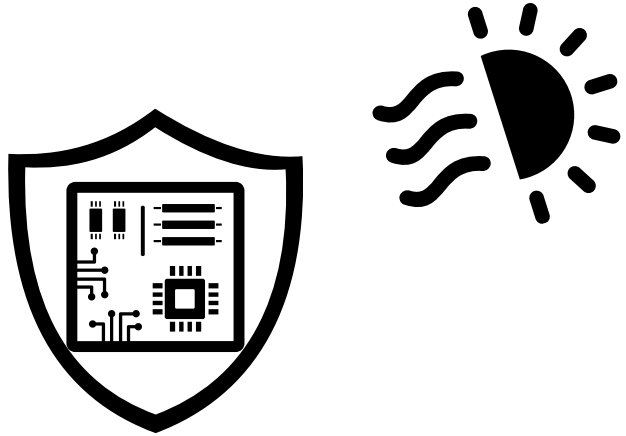
  → time-intensive
- Fast testing in-situ (by uploading)
  - might endanger mission

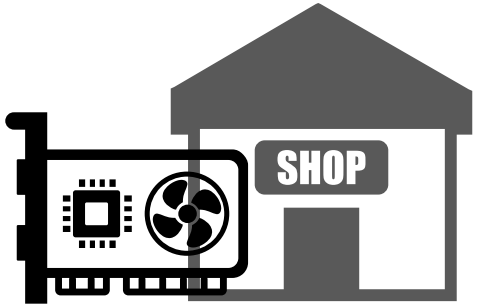


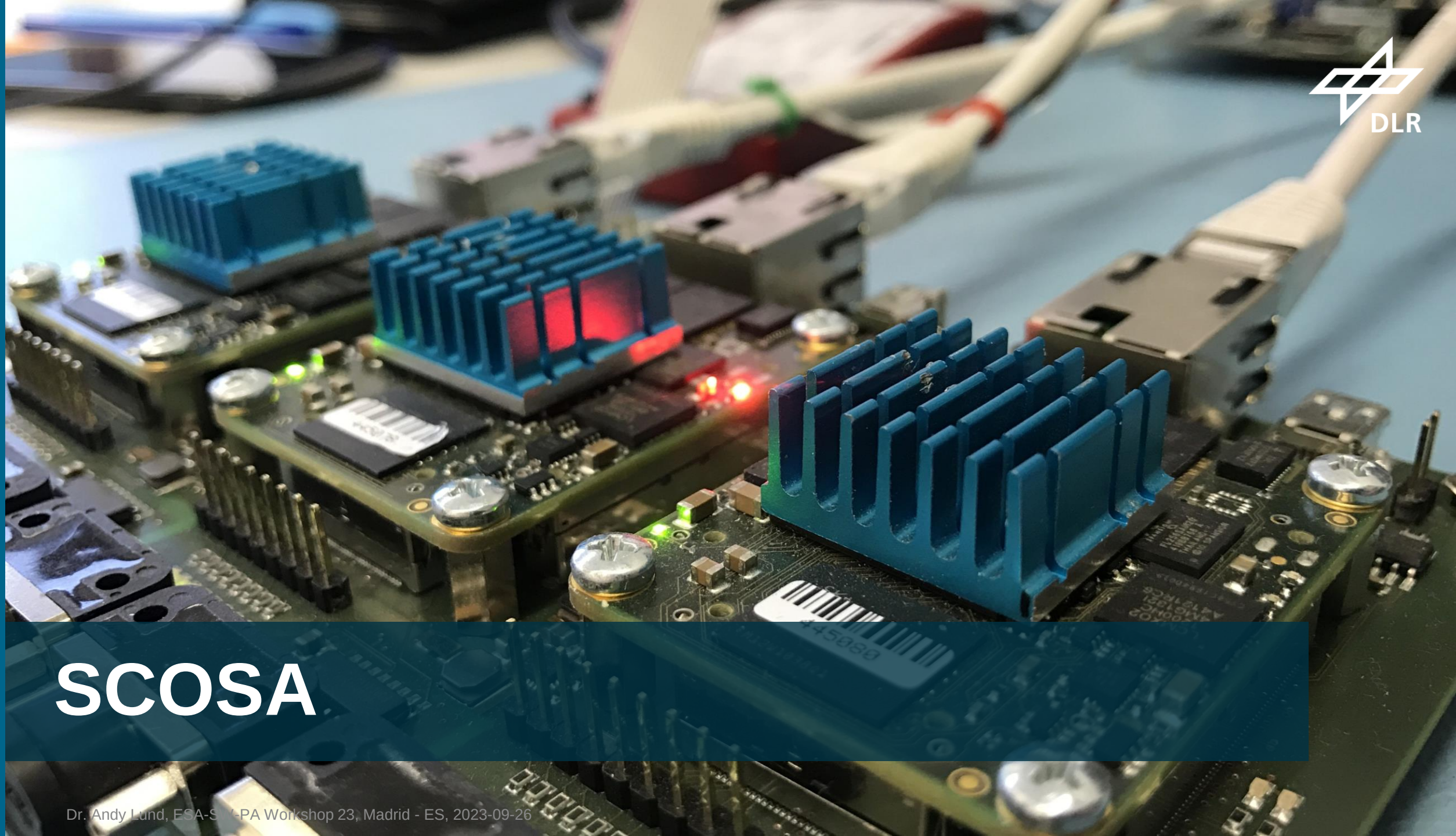
Let‘s test some software in space

- Satellite OBCs are often
  - custom development
    - radiation-hardened
  - single node
  - inaccessible eco systems

  → prototyping hard
- Cubesats include more and more
  - commercial-off-the-shelf (COTS)
  - ARM core
  - Separate subsystem
  - Single SoC





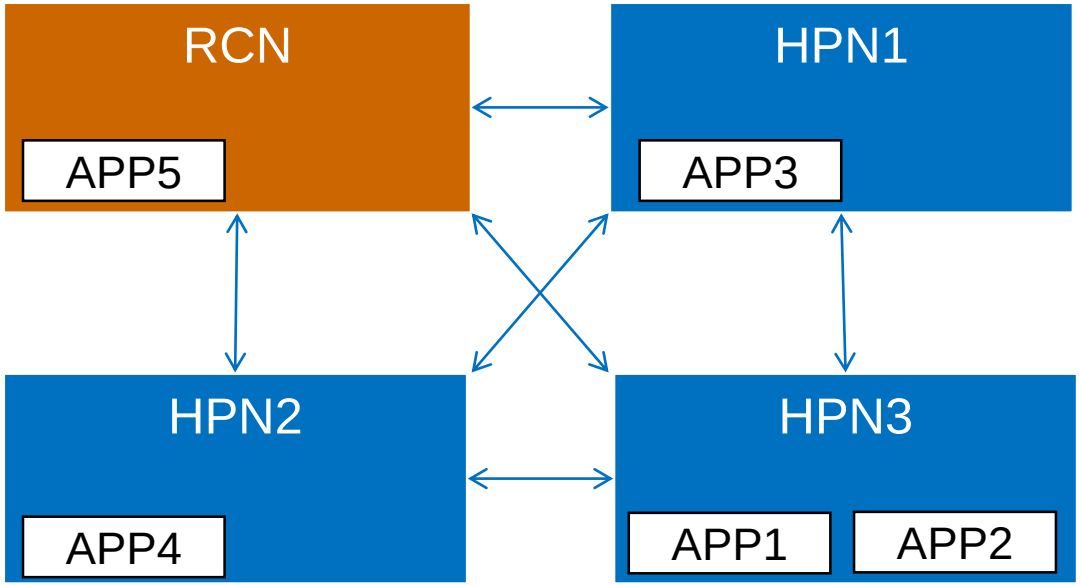


SCOSA

Scalable On-Board Computing for Space Avionics (ScOSA)

Combining space-qualified hardware with COTS (Commercial-off-the-shelf) components

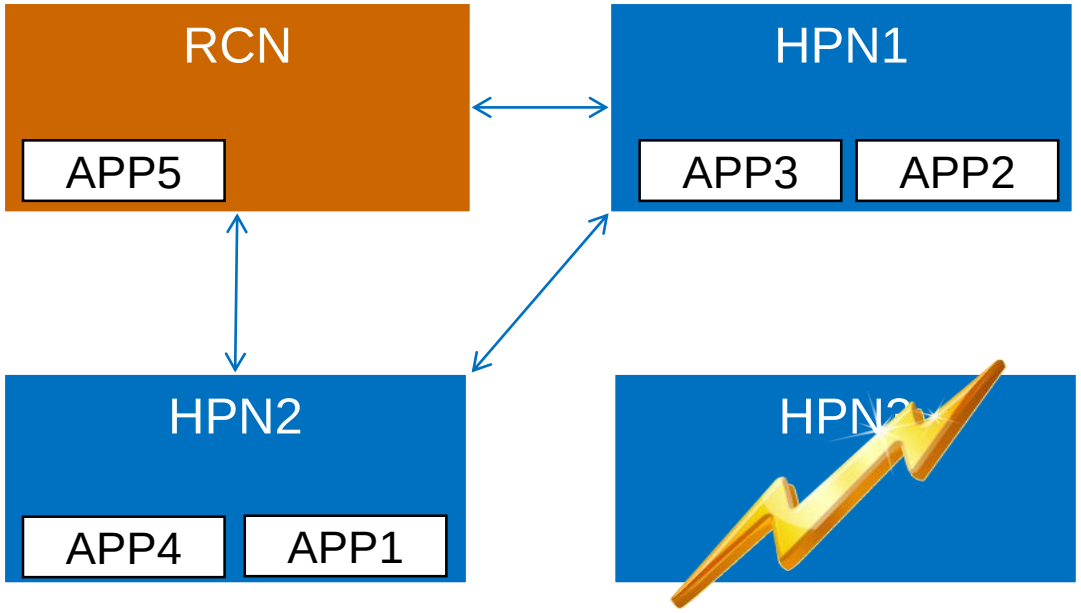
Reliable Computing nodes (RCNs)	• Mission-critical tasks • Fallback
High-Performance nodes (HPNs) (Zynq-7000)	• High-Performance tasks
In case of node-failures:	• Reordering of the task mapping



Scalable On-Board Computing for Space Avionics (ScOSA)

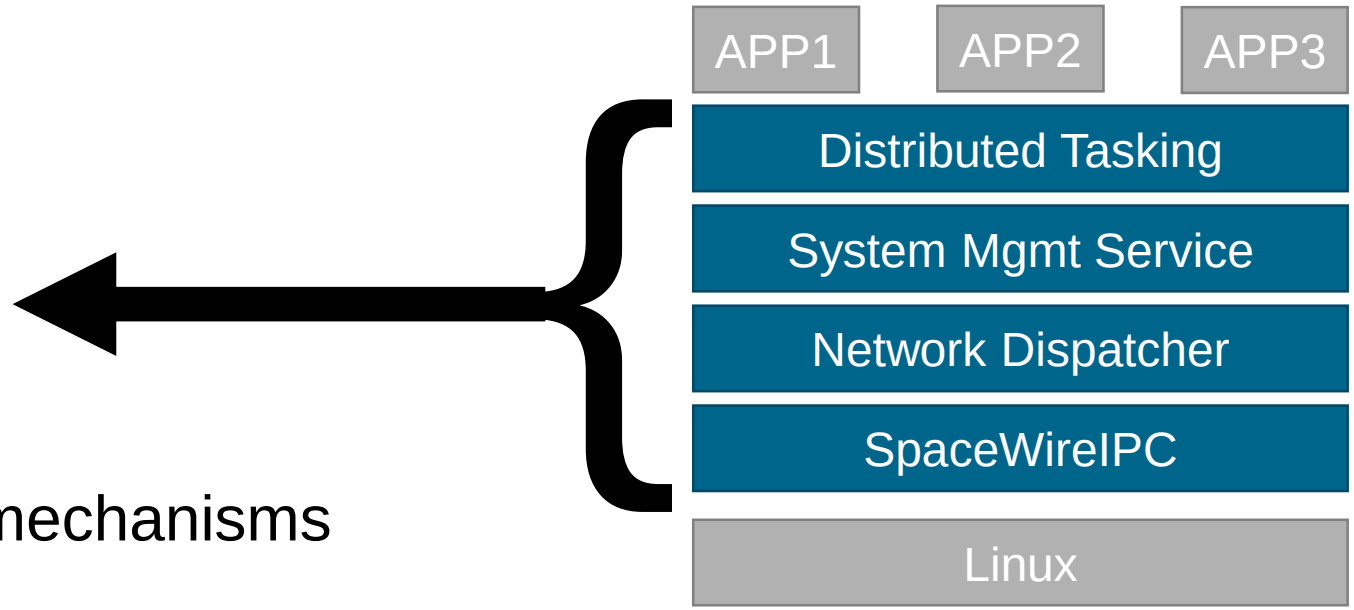
Combining space-qualified hardware with COTS (Commercial-off-the-shelf) components

Reliable Computing nodes (RCNs)	• Mission-critical tasks • Fallback
High-Performance nodes (HPNs) (Zynq-7000)	• High-Performance tasks
In case of node-failures:	• Reordering of the task mapping



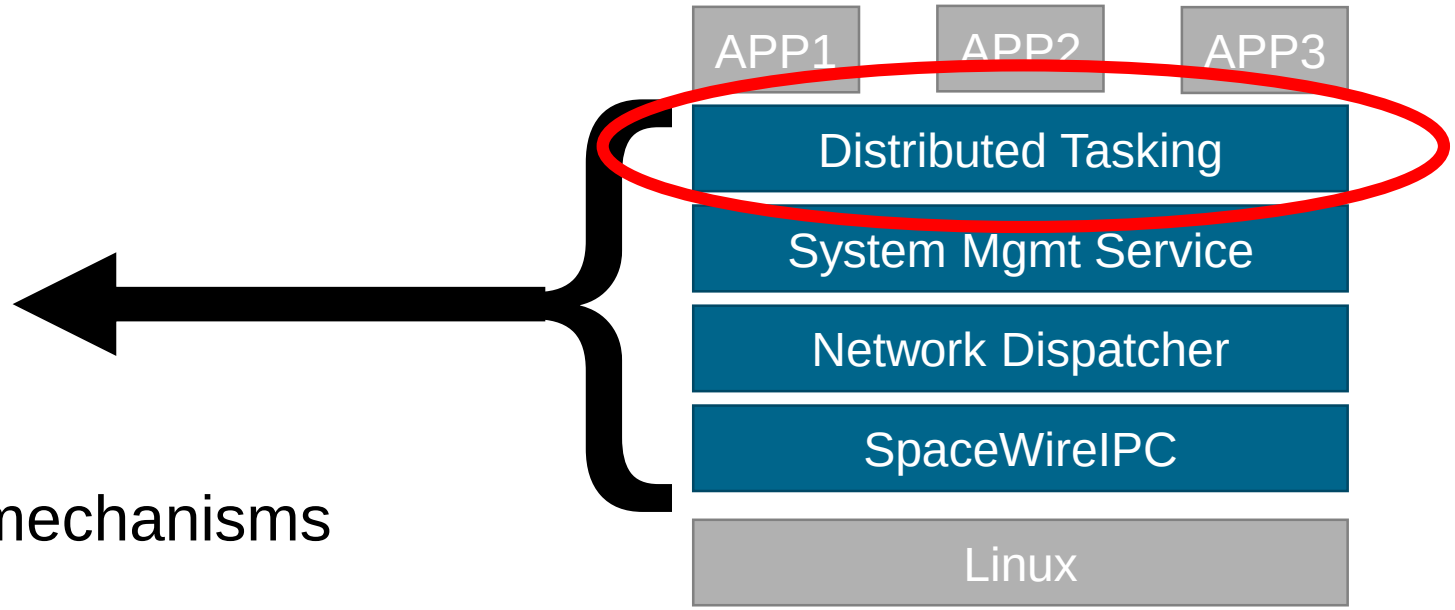
ScOSA: Middleware

- Abstraction of distributed architecture
  - Linux & RTEMS
  - SpaceWire & Ethernet
- Consists of several layers
- Enables Fault-Tolerance mechanisms
  - Reconfiguration
  - Reliable messaging
  - Voting (TMR)



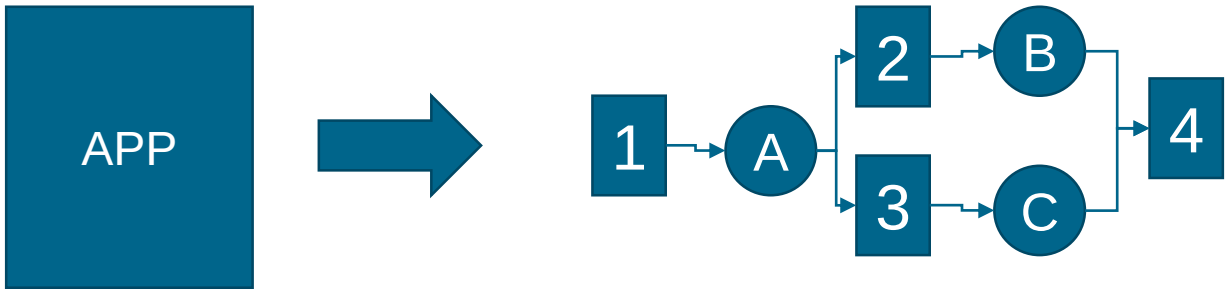
ScOSA: Middleware

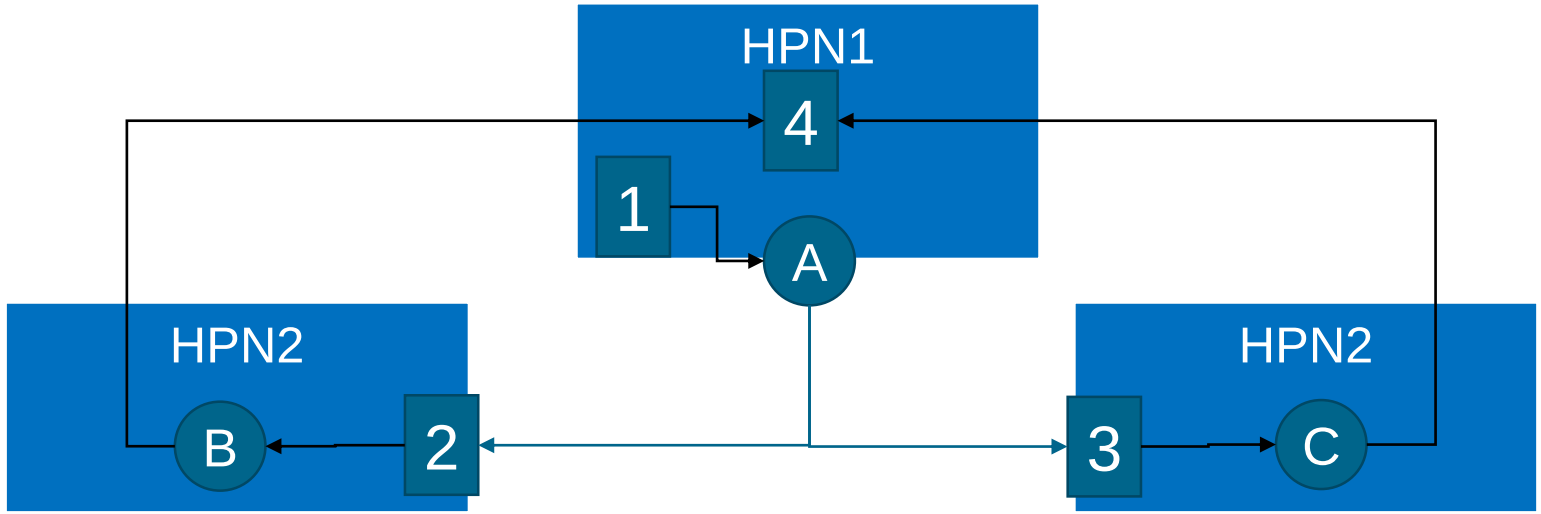
- Abstraction of distributed architecture
  - Linux & RTEMS
  - SpaceWire & Ethernet
- Consists of several layers
- Enables Fault-Tolerance mechanisms
  - Reconfiguration
  - Reliable messaging
  - Voting (TMR)



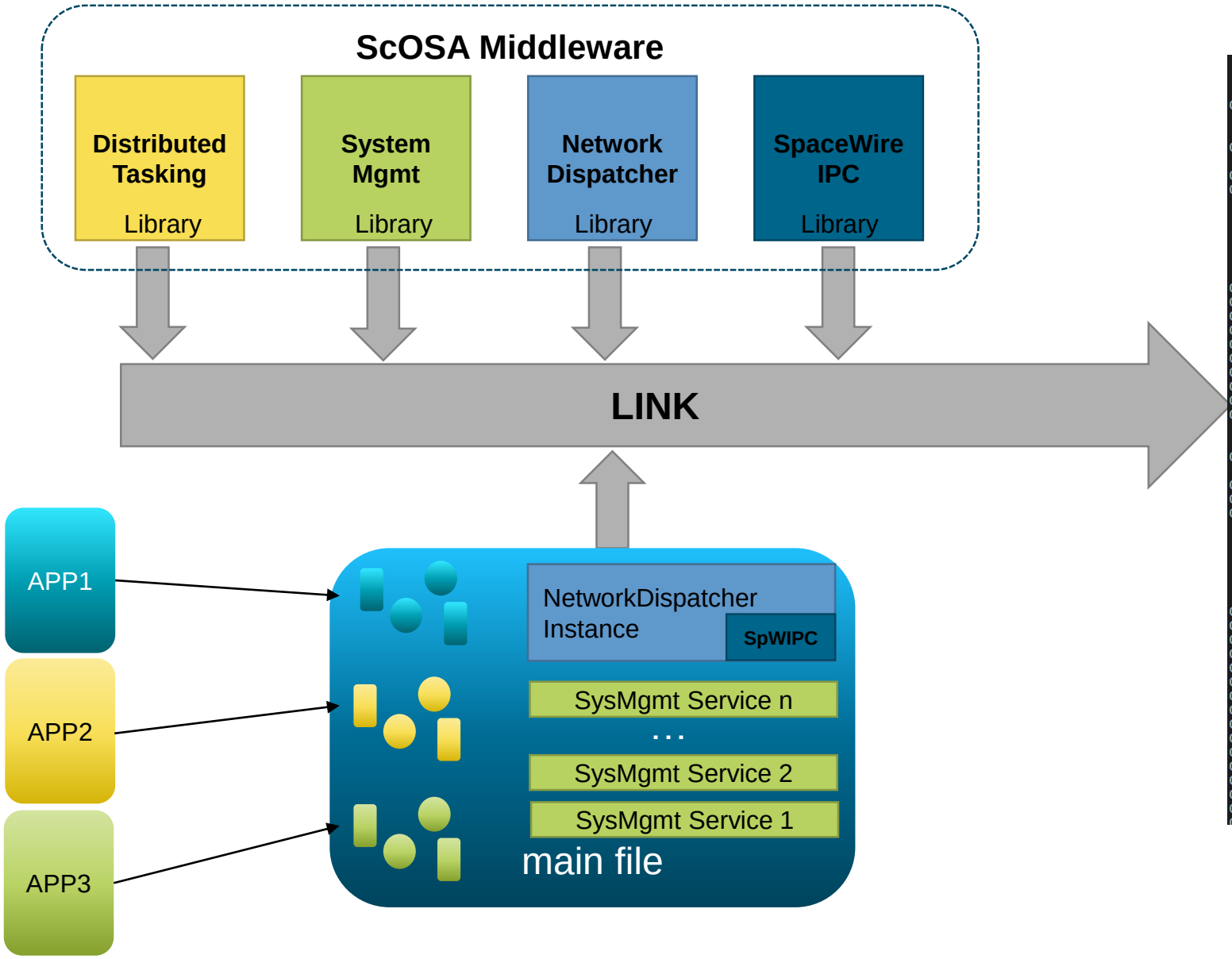
Excursion: Distributed Tasking Framework

- APP → set of inter-connected tasks and channels → data flow oriented



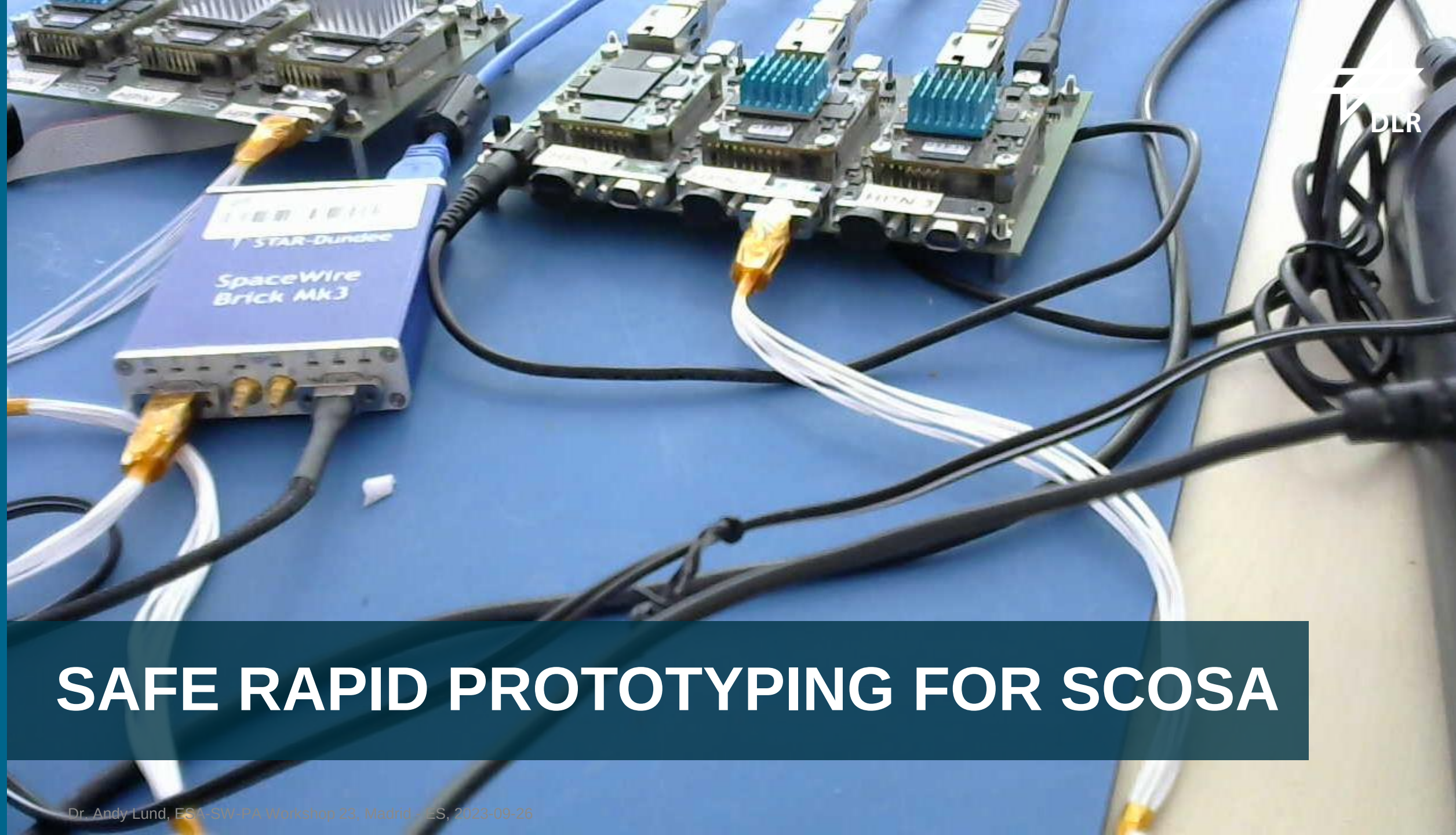


Building the System




```
                 U abort@@GLIBC_2.2.5
                 U __assert_fail@@GLIBC_2.2.5
                 U bind@@GLIBC_2.2.5
0000000000280040 B __bss_start
                 U clock_gettime@@GLIBC_2.2.5
                 U clock_nanosleep@@GLIBC_2.2.5
0000000000042130 T clockThread
                 U close@@GLIBC_2.2.5
0000000000280270 b completed.7698
000000000035d800 B consoleAppender
                 U __cxa_atexit@@GLIBC_2.2.5
                 w __cxa_finalize@@GLIBC_2.2.5
                 U __cxa_guard_abort@@CXXABI_1.3
                 U __cxa_guard_acquire@@CXXABI_1.3
                 U __cxa_guard_release@@CXXABI_1.3
                 U __cxa_pure_virtual@@CXXABI_1.3
0000000000280000 D __data_start
0000000000280000 W data_start
0000000000008070 t deregister_tm_clones
0000000000008100 t __do_global_dtors_aux
000000000027dc58 d __do_global_dtors_aux_fini_array_entry
0000000000280008 D __dso_handle
0000000000280018 V DW.ref.__gxx_personality_v0
000000000027fa08 d _DYNAMIC
0000000000280040 D _edata
000000000035d9a0 B _end
                 U exit@@GLIBC_2.2.5
                 U fclose@@GLIBC_2.2.5
00000000000671f4 T _fini
                 U fopen@@GLIBC_2.2.5
0000000000008140 t frame_dummy
000000000027dbc0 d __frame_dummy_init_array_entry
000000000007b85c r __FRAME_END__
                 U __fread_chk@@GLIBC_2.7
                 U fread@@GLIBC_2.2.5
                 U fseek@@GLIBC_2.2.5
                 U ftell@@GLIBC_2.2.5
                 U fwrite@@GLIBC_2.2.5
                 U gettimeofday@@GLIBC_2.2.5
000000000027fc38 d _GLOBAL_OFFSET_TABLE_
0000000000007f80 t _GLOBAL__sub_I_consoleAppender
0000000000007be0 t _GLOBAL__sub_I_node1
0000000000007f50 t _GLOBAL__sub_I__ZN13spacewire_ipc12Transmission11isBigEndianEv
0000000000007cb0 t _GLOBAL__sub_I__ZN17system_management15ManagementTasksC2EhPN7Task
0000000000007d10 t _GLOBAL__sub_I__ZN17system_management17ManagementHandler25registe
0000000000007dd0 t _GLOBAL__sub_I__ZN17system_management18SystemAlertServiceC2EPNS_
0000000000007e60 t _GLOBAL__sub_I__ZN17system_management19ConfigurationMatrix11addRe
0000000000007ce0 t _GLOBAL__sub_I__ZN17system_management19SystemConfigurationC2EPN1
0000000000007e30 t _GLOBAL__sub_I__ZN17system_management20CheckpointingServiceC2EPNS
0000000000007e00 t _GLOBAL__sub_I__ZN17system_management20ReintegrationServiceC2EPNS
0000000000007da0 t _GLOBAL__sub_I__ZN17system_management20SystemMonitorServiceC2EPNS
0000000000007d70 t _GLOBAL__sub_I__ZN17system_management22ReconfigurationManager16is
0000000000007d40 t _GLOBAL__sub_I__ZN17system_management22ReconfigurationServiceC2EP
0000000000007f20 t _GLOBAL__sub_I__ZN18network_dispatcher17NetworkDispatcherC2Eh
0000000000008010 t _GLOBAL__sub_I__ZN7outpost4rtos6Thread7wrapperEPv
```

One process to rule them all



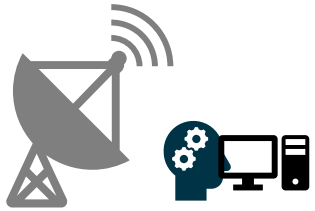
SAFE RAPID PROTOTYPING FOR SCOSA

In-flight Updates

- In-orbit
  - Development continues
  - New Apps shall be tested

In-flight Updates

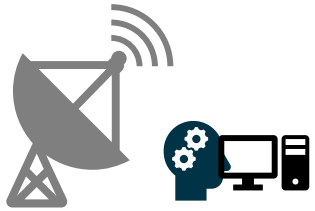
- In-orbit
  - Development continues
  - New Apps shall be tested
- We need:
  - Upload mechanism



In-flight Updates

- In-orbit
  - Development continues
  - New Apps shall be tested

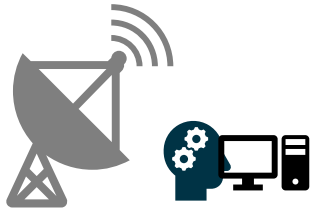
- We need:
  - Upload mechanism ✅
  - An OS which easily supports replacing of binary
    - Linux ✅



In-flight Updates

- In-orbit
  - Development continues
  - New Apps shall be tested

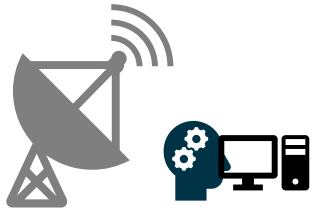
- We need:
  - Upload mechanism ✅
  - An OS which easily supports replacing of binary
    - Linux ✅
  - A fallback strategy ~



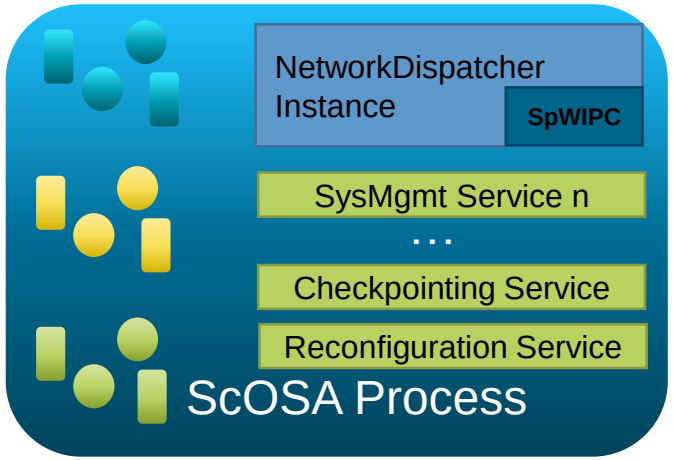
In-flight Updates

- In-orbit
  - Development continues
  - New Apps shall be tested

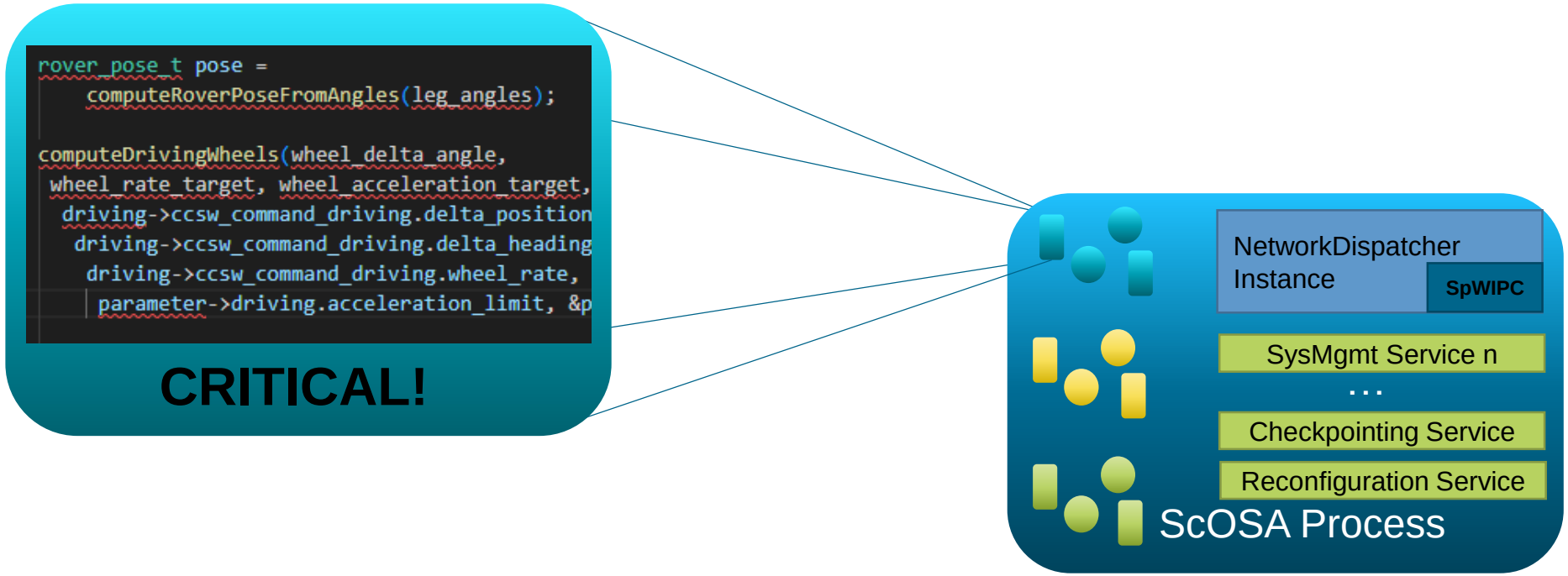
- We need:
  - Upload mechanism ✓
  - An OS which easily supports replacing of binary
    - Linux ✓
  - A fallback strategy ~
  - Protect the middleware and other apps ✗



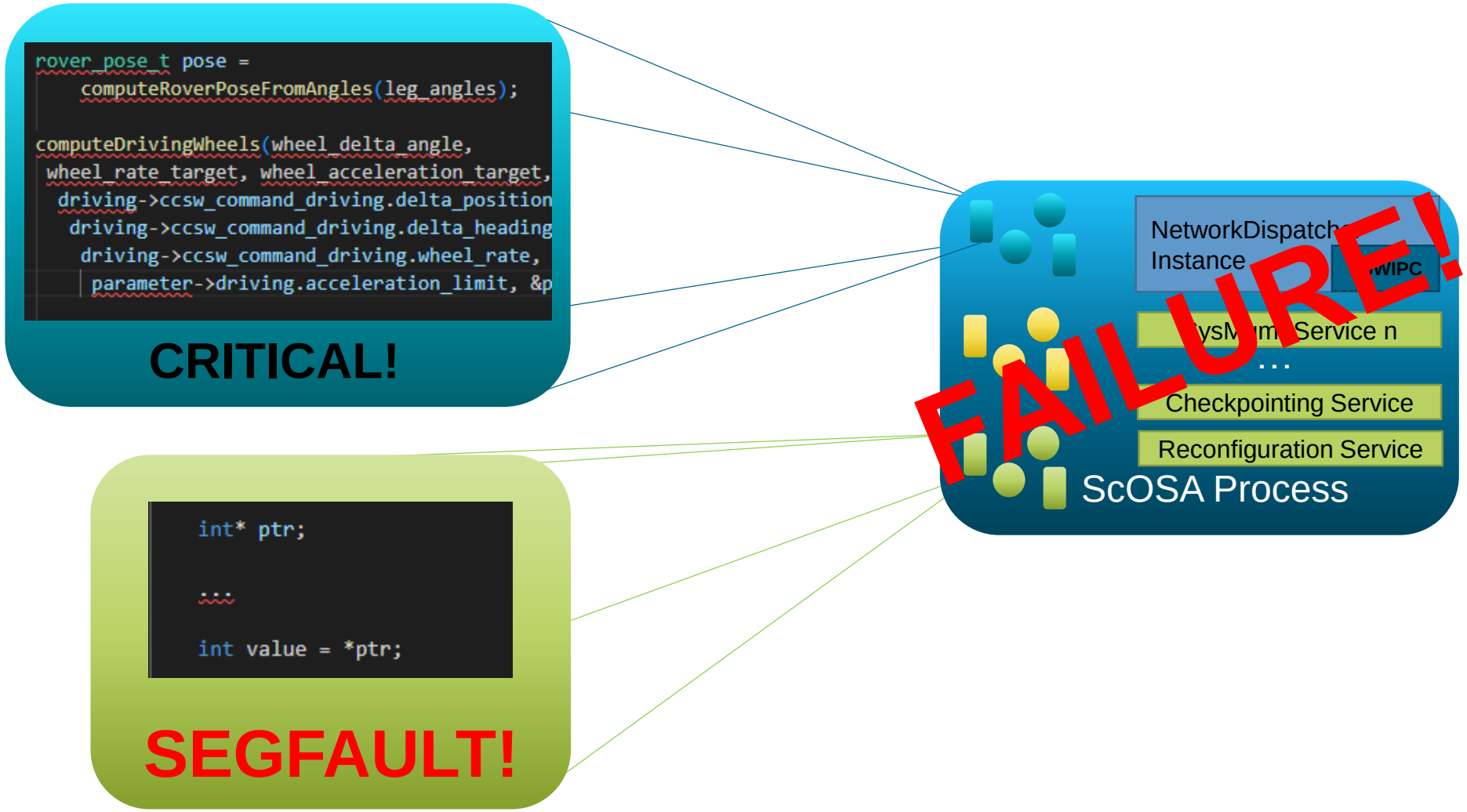
What might be the problem?



What might be the problem?

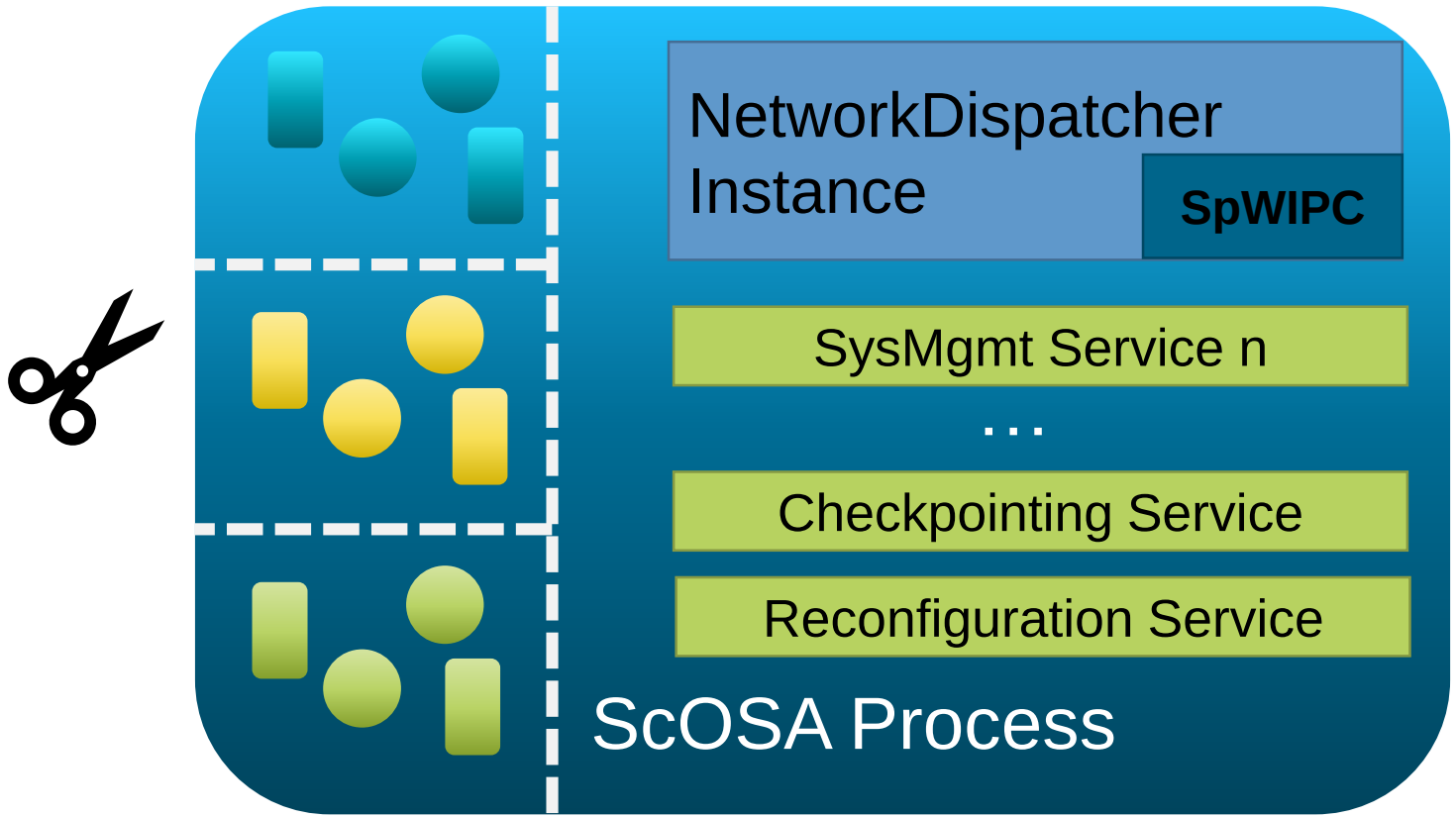


What might be the problem?

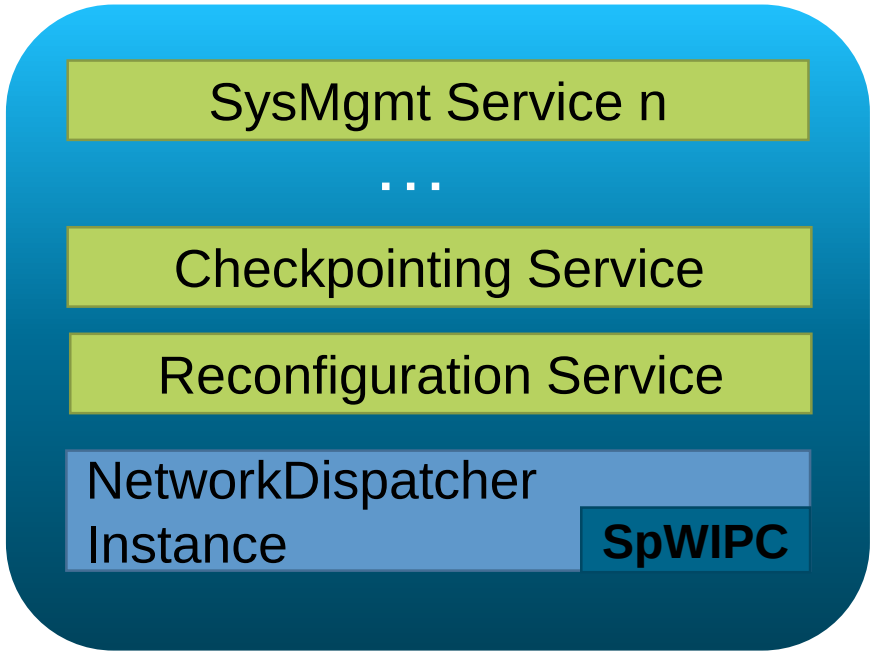


One way to solve this

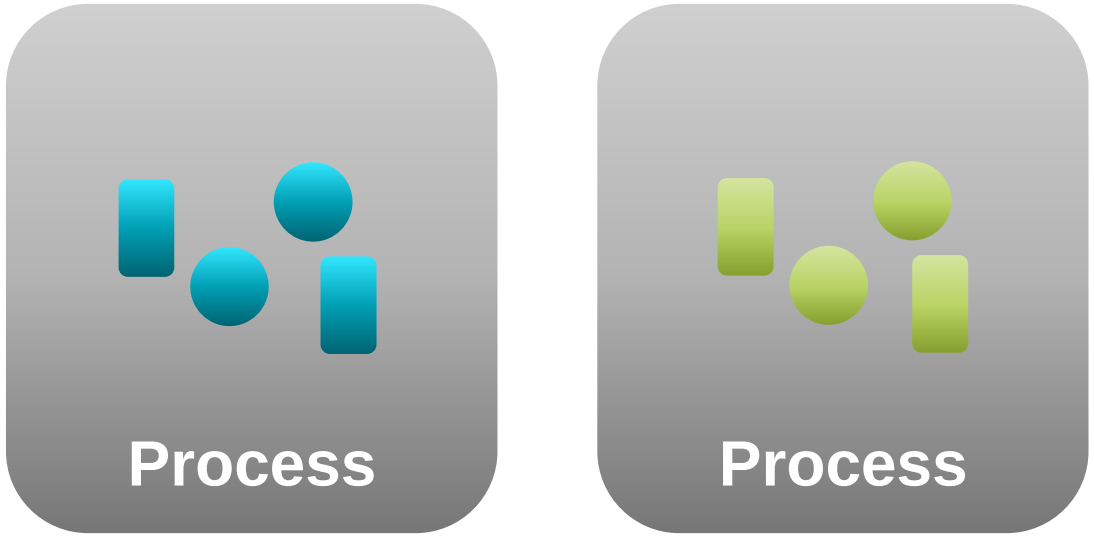
Divide & Conquer → Separation & Isolation



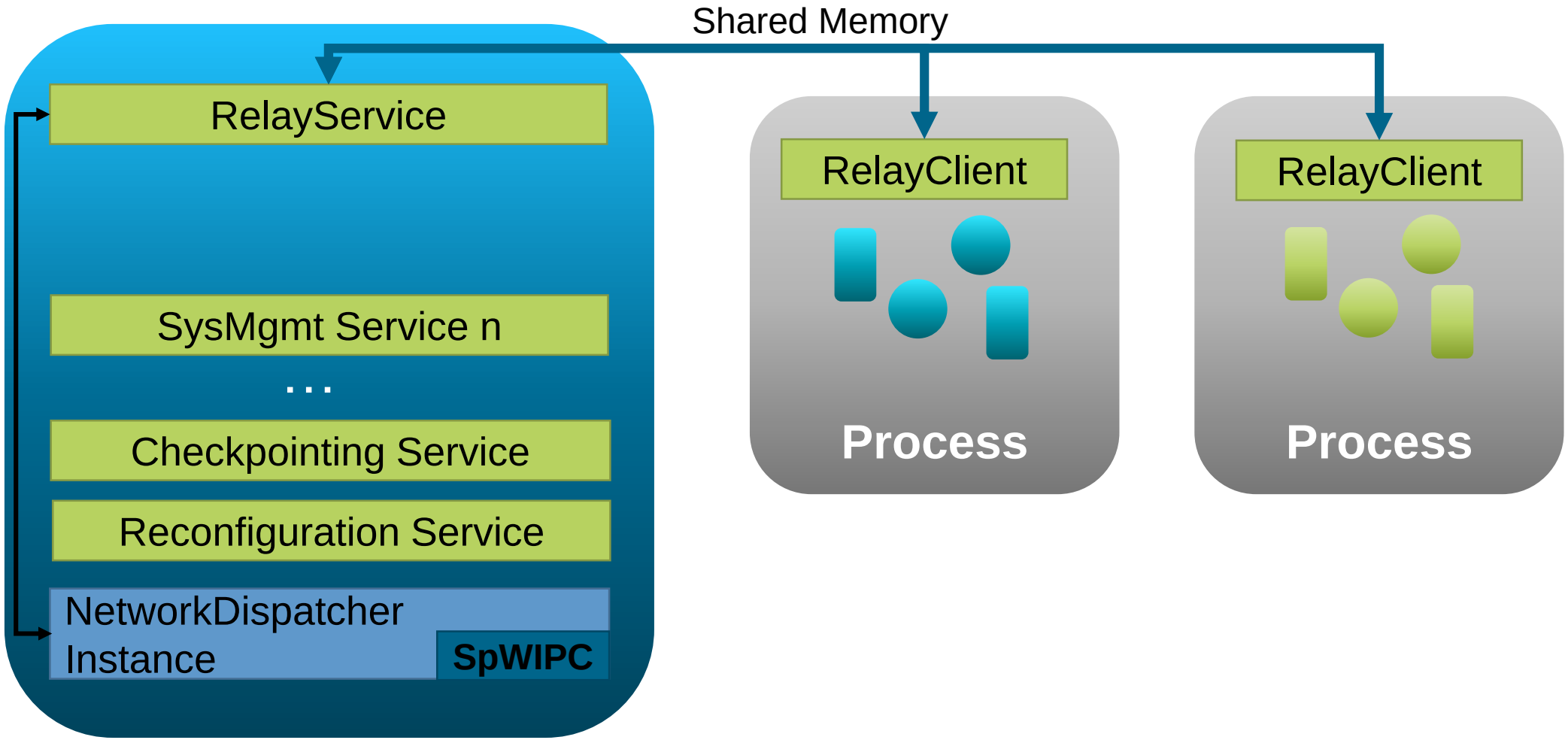
Separation of Apps



Dynamic spawning during reconfiguration

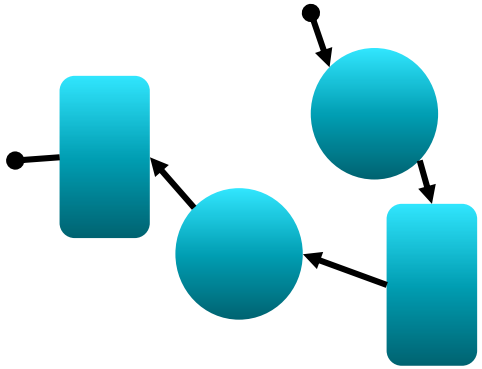


Separation of Apps



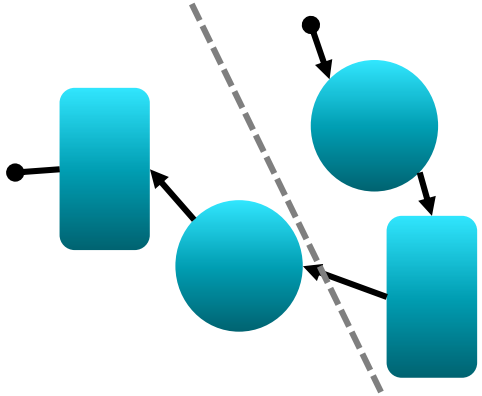
Parallelization

- Underlying *Distributed Tasking Framework* structure is still working



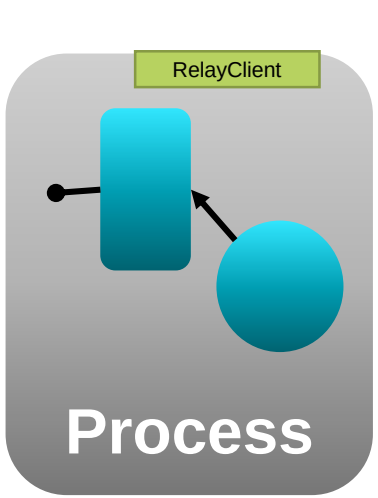
Parallelization

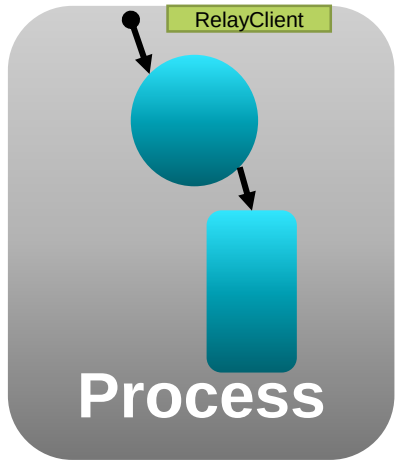
- Underlying *Distributed Tasking Framework* structure is still working



Parallelization

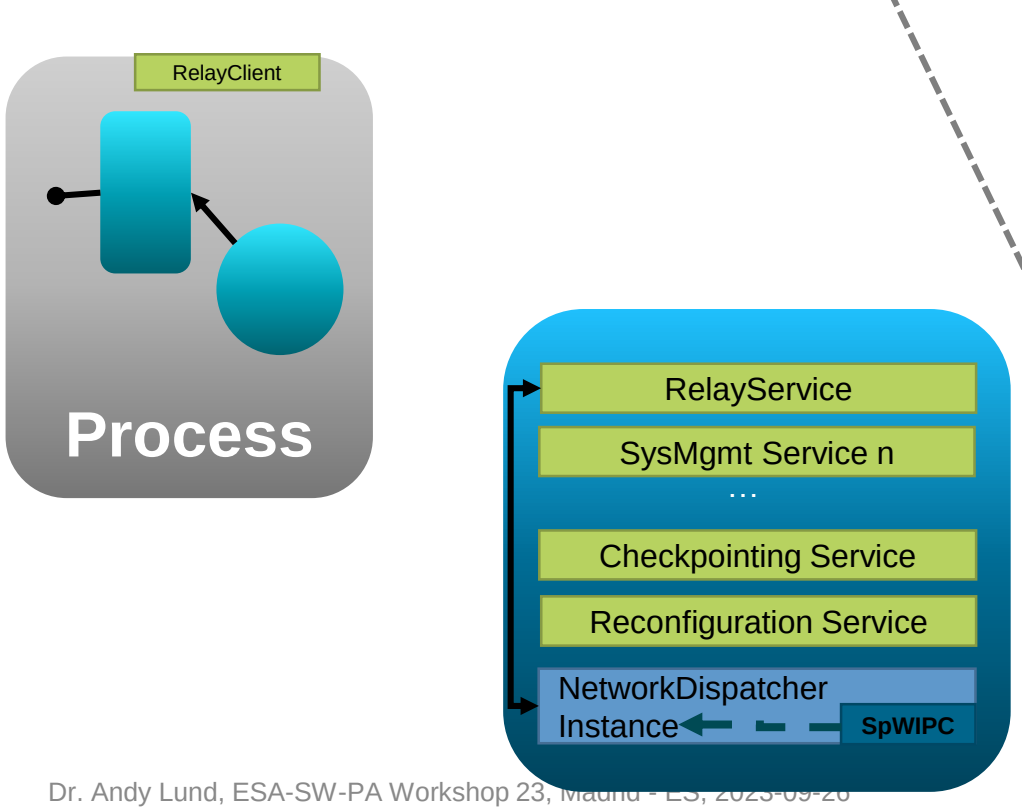
- Underlying *Distributed Tasking Framework* structure is still working

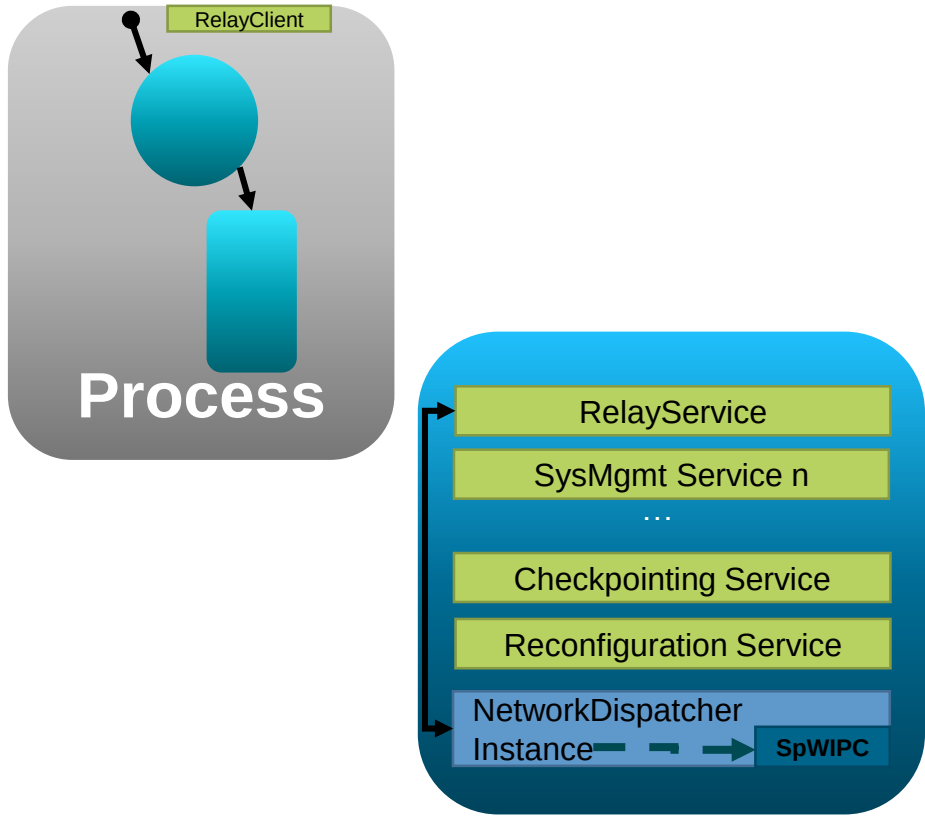




Parallelization

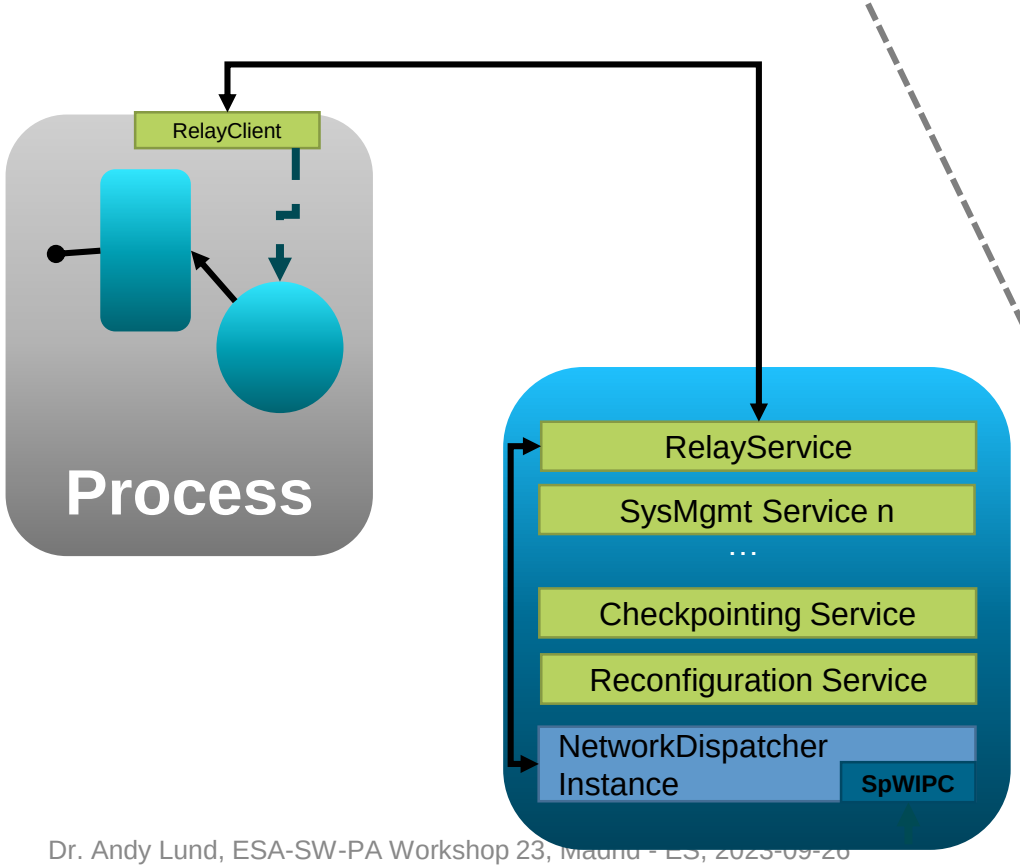
- Underlying *Distributed Tasking Framework* structure is still working

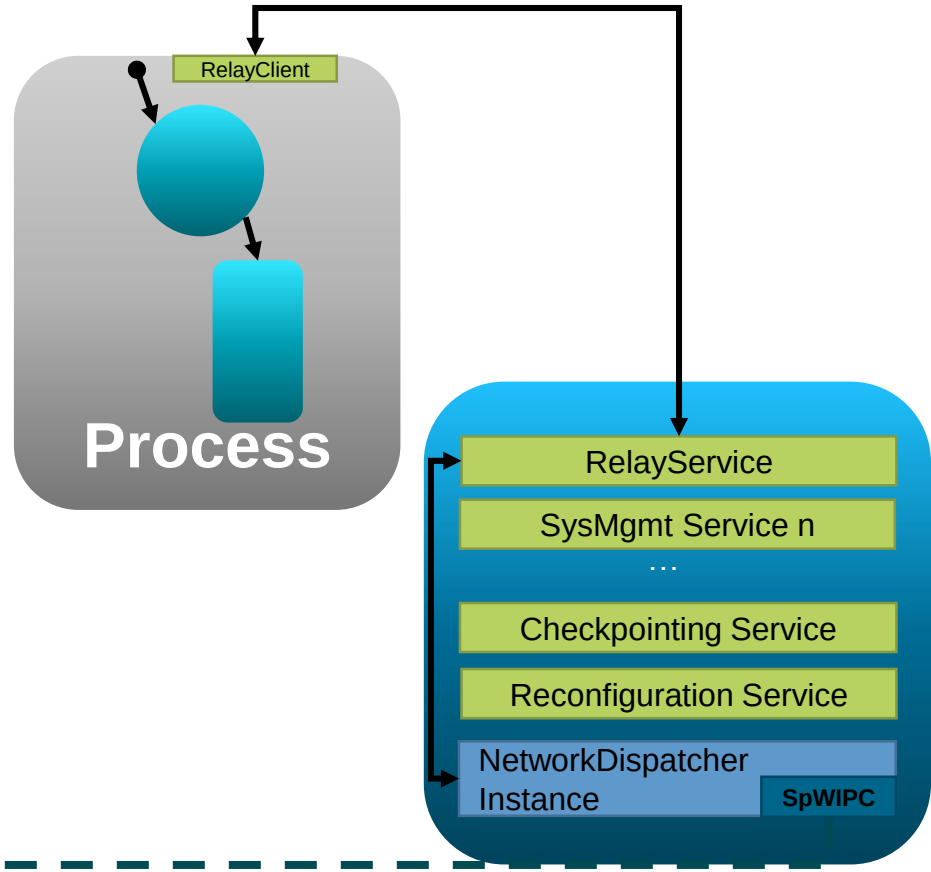




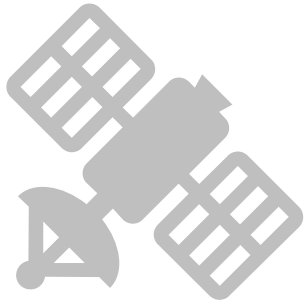
Parallelization

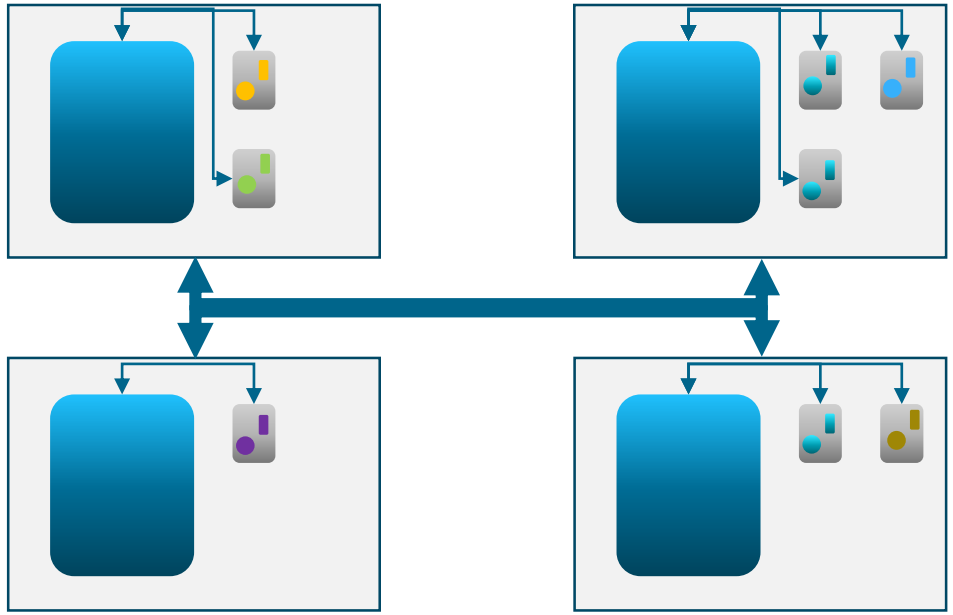
- Underlying *Distributed Tasking Framework* structure is still working

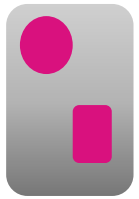




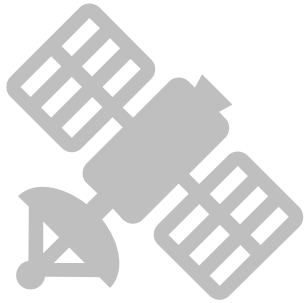
Wrap-up

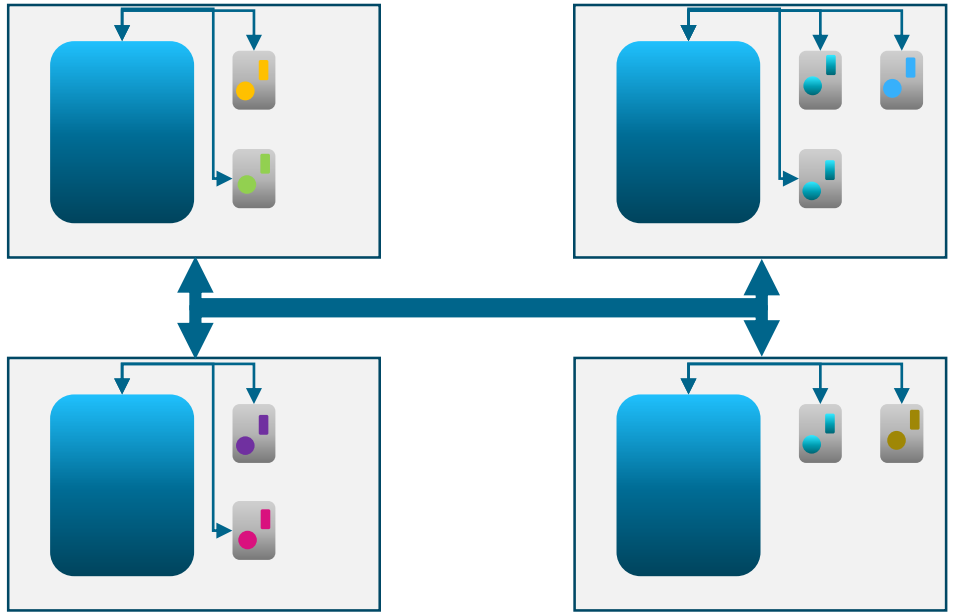




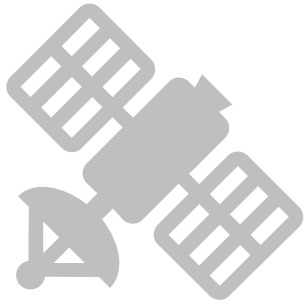


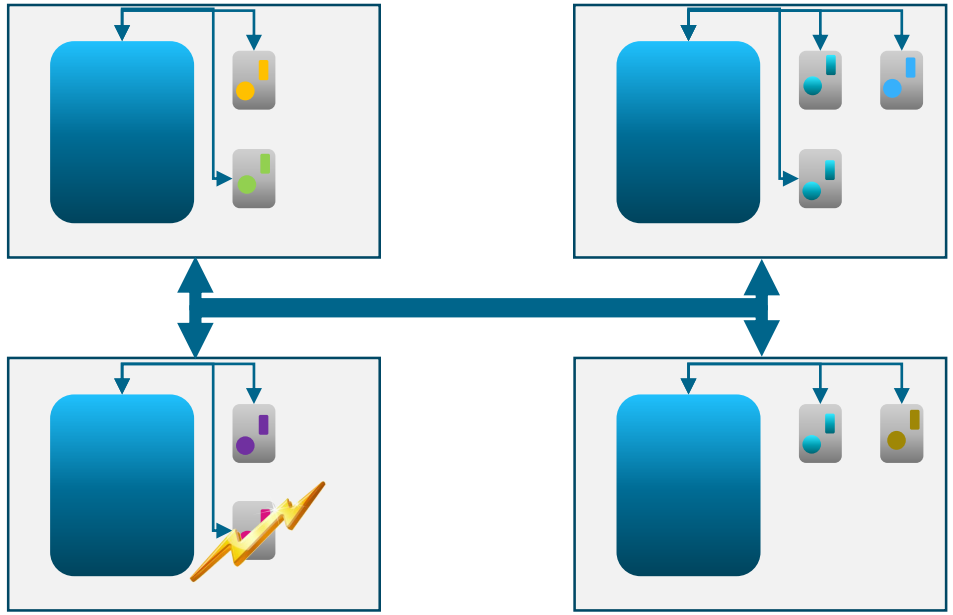
Wrap-up



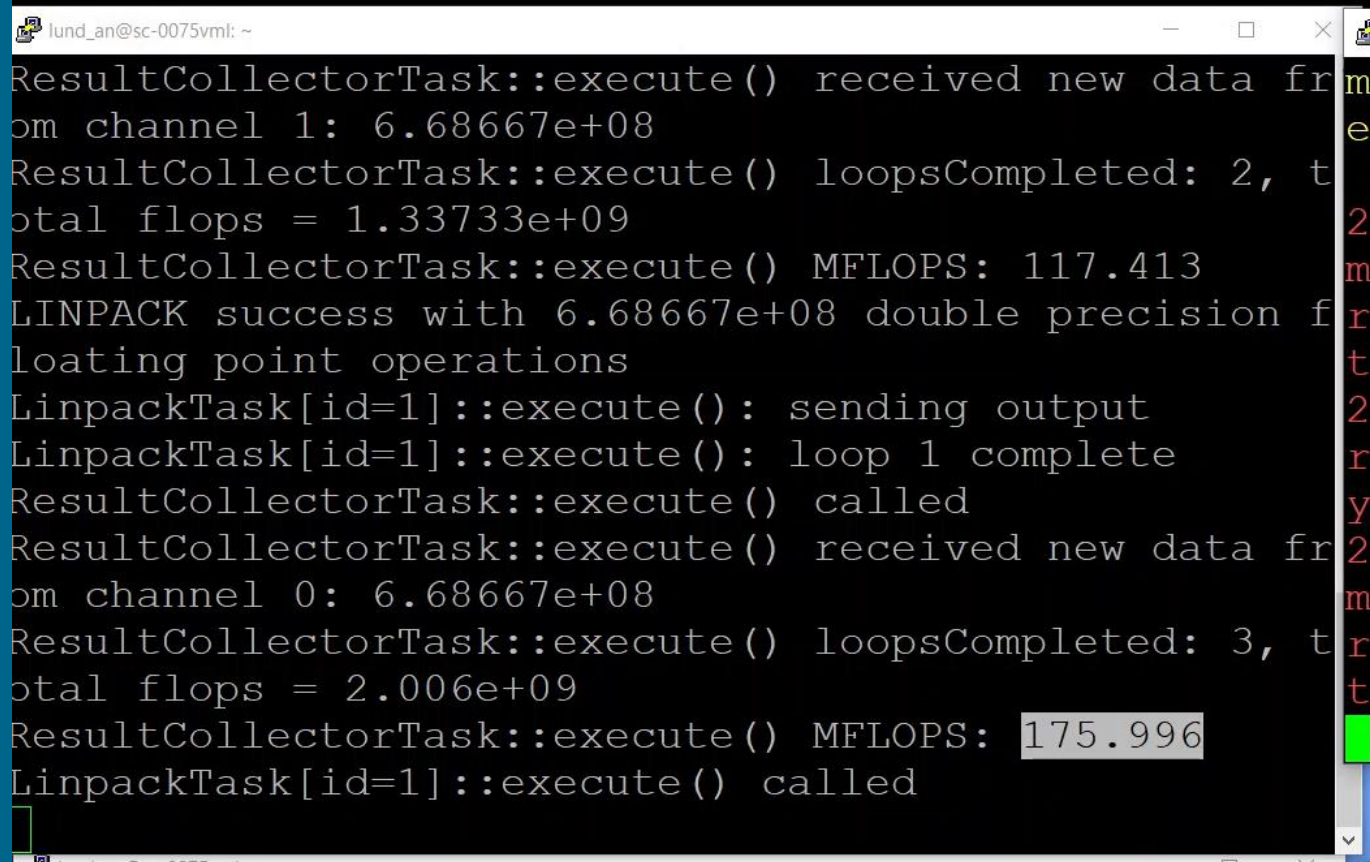


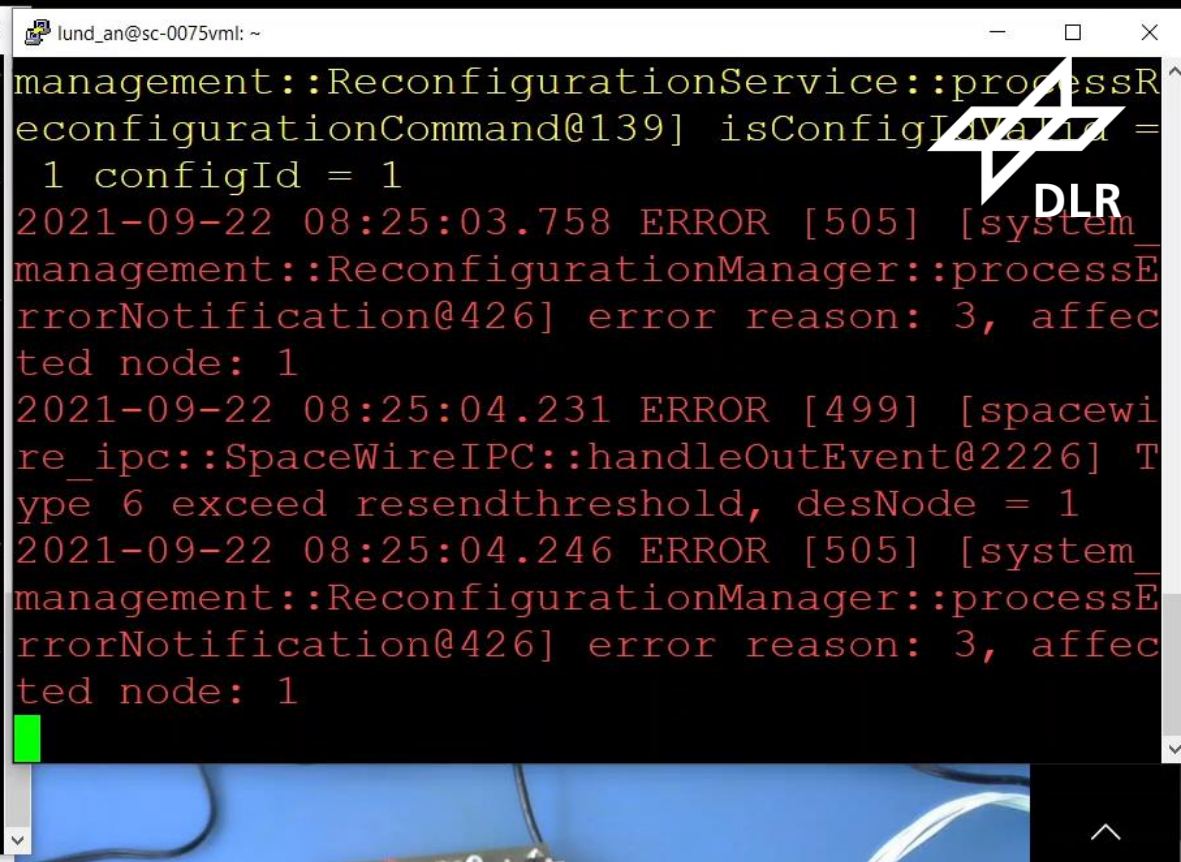
Wrap-up

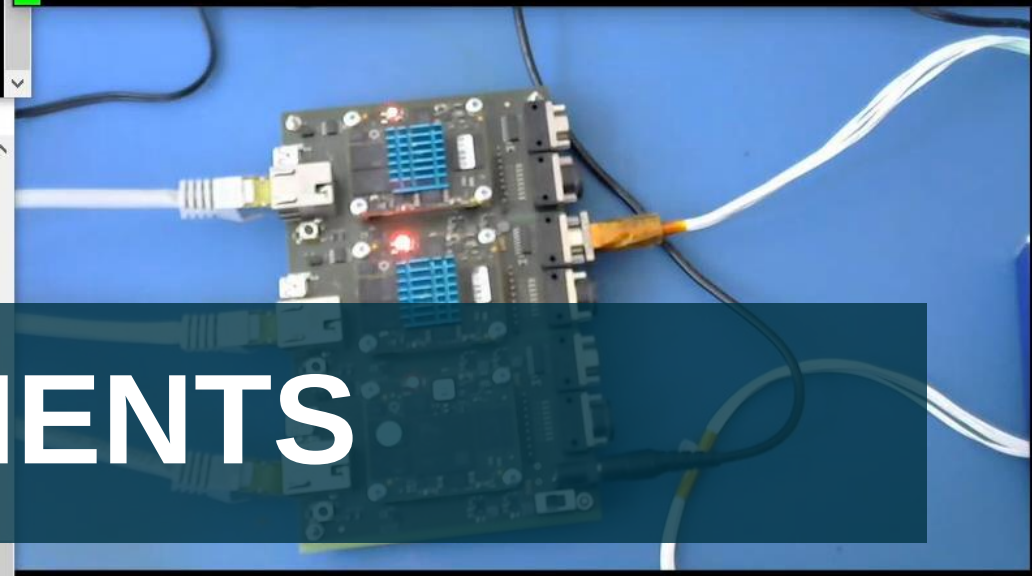




Neither middleware nor other apps are endangered







FUTURE IMPROVEMENTS

CGroups

- Control Groups
  - Part of the linux kernel
  - Since 2008

- Processes get ordered into groups
  - Groups can be limited in resources:
    - Memory
    - CPU
    - I/O

→ Giving ScOSA middleware the ability to further control the apps

Summary & Questions

- ScOSA OBC Architecture
  - Middleware with autonomous reconfiguration
  - Going into space 2024
  - One binary including apps and middleware services
- Enabling rapid prototyping and mixed-criticality
  - By dynamically spawning processes
  - Connected with the middleware by shared memory
  - Keeping the property of distributed parallelization
- CGroups

Summary & Questions

- ScOSA OBC Architecture
  - Middleware with autonomous reconfiguration
  - Going into space 2024
  - One binary including apps and middleware services

- Enabling rapid prototyping and mixed-criticality
  - By dynamically spawning processes
  - Connected with the middleware by shared memory
  - Keeping the property of distributed parallelization

Thanks!
Andreas.Lund@dlr.de

- CGroups

BACKLOG

```
bool
SystemConfiguration::processReconfigurationService(scosa_system::config_t configId,
                                                   scosa_system::channelId_t minChannel,
                                                   scosa_system::channelId_t maxChannel)
{
    network_dispatcher::RoutingTable* p_routingTable =
            &(configurationSet.configurations[configId].routingTable);
    Configuration config = configurationSet.configurations[configId];

    for (uint i = 0; i < network_dispatcher::processMaxNum; i++)
    {
        if (taskingTable.p_process[i] != nullptr)
        {
            if (taskingTable.p_process[i]->isRunning() && !p_routingTable->processRoutingInfo[i].isOnNode[getNodeId()])
            {
                LOGD << "Killing process "<< i;
                taskingTable.p_process[i]->killProc();
                m_relayService->closeIPC(taskingTable.p_process[i]->getId());       You, 2 weeks ago • added relayService …
            }
            else if (!taskingTable.p_process[i]->isRunning() && p_routingTable->processRoutingInfo[i].isOnNode[getNodeId()]){
                LOGD << "Spawning process "<< i << " with configId " << configId;
                taskingTable.p_process[i]->create(configId);
                m_relayService->createIPC(taskingTable.p_process[i]->getId());
            }

        }
    }
}
```

```cpp
bool Process::create(scosa_system::config_t configId)
{
    m_processId = fork();

    if(m_processId < 0){
        return false;
    } else if (m_processId == 0) {
```

```cpp
static char *newenviron[] = { NULL };
execve(m_execPath.c_str(), newargv, newenviron);
exit(EXIT_FAILURE);
```

ScOSA Demonstration

- Demonstration mission 2024 (CAPTn-1)
  - Evaluating the behaviour in space
    - Reconfiguration
    - Performance
  - Demonstrating typical space appliactions
    - ODARIS
      - Earth observation information and alarm service
    - ORS
      - Simulating an on-board rendez-vous navigation
    - Image Compression by co-processor
    - SEU detection