# AADL - Reverse-Engineering of a satellite wheel

***Date: 16/02/2016***

***Developed by:***

***Roland SCHURIG***

***Prerana SHAMSUNDAR PUNJABI***

# Table of Contents

# Executive Summary

The assignment aimed at designing an AADL model for a satellite wheel as a complete control/command system that is used to correct the attitude of a satellite

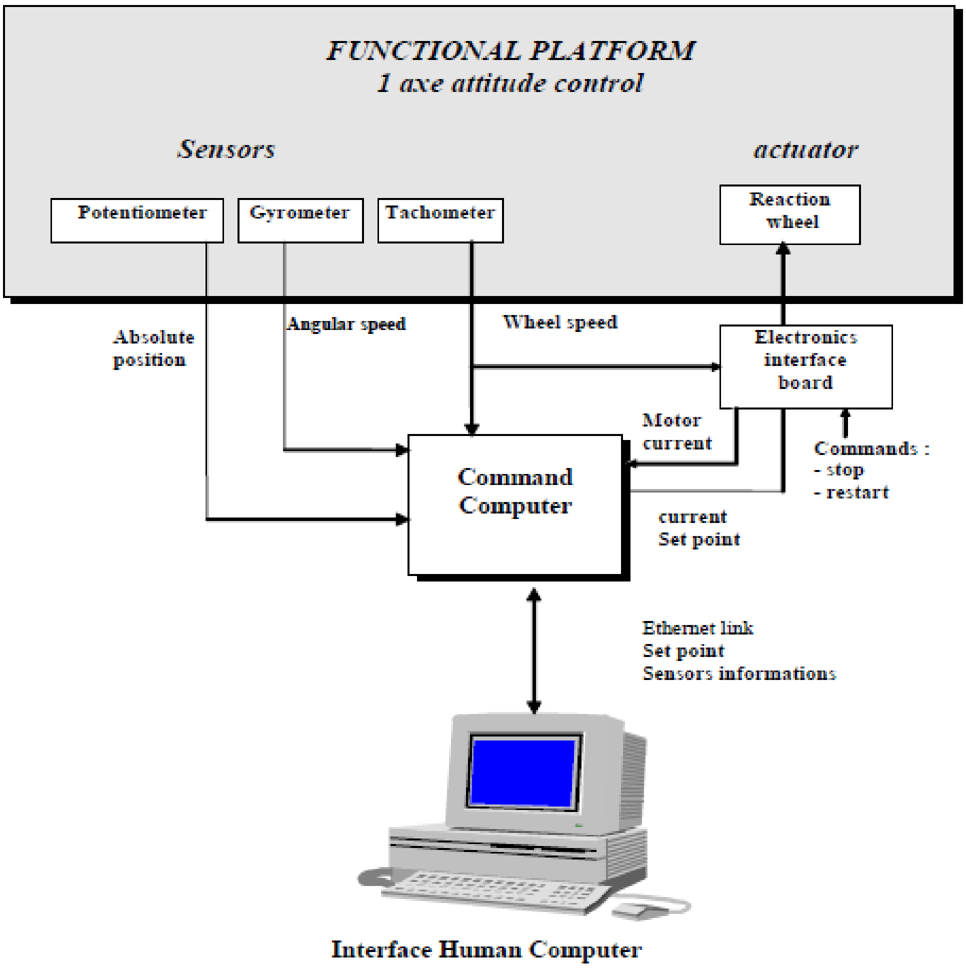
The system to be modeled comprises of:

- Mechanical devices - a sensors and a motor
- A graphical user interface for receiving information from the wheel, and plotting some values
- An onboard processor that implements control laws to pilot the motor

The Reaction Wheel that is driven by a DC Motor to speed up or speed down and generate torques.

An embedded computer is used to execute the command laws to control the wheel.

A Human Machine Interface (HMI) is used to interact with the wheel and observe the curves that are displayed.



# 1 Initial Model of the Onboard Processor

**Design Description**:

The onboard processor is designed with respect to the specification of the following threads:

| Tasks | Functions | Activation in us | Deadline in us | WCET in us | Priority | Utilization (WCET/Period) |
|---|---|---|---|---|---|---|
| F1 - TDialog | To manage request | 100000 | 100000 | 2000 | 1 | 0.02 |
| F2 - TScheduler | To manage duration and periods | 1000 | 1000 | 200 | 4 | 0.2 |
| F3 - TActuator | To manage law and set point | 10000 | 10000 | 1000 | 2 | 0.1 |
| F4 - TAquire | To Acquire curves information | 2000 | 2000 | 300 | 3 | 0.15 |

The Thread TDialog is aperiodic but for the worst case schedulability analysis we need to consider it as periodic. A period and deadline of 100ms is assigned to this thread as it receives requests from the user (as a natural latency exits between any 2 user inputs).

The priorities of the threads are assigned as stated above. The Scheduler has the highest priority: '5' and tDialog has the lowest priority: '1'. The priorities were assigned by taking into account the Worst case execution time of each thread: The task with lowest WCET was assigned with the highest priority.

The WCET is set as the upper bound in order to model for the worst case scenario this would also to make the response time more deterministic.

A **Process** `SatComp` schedules the above stated threads using a fixed priority scheduling policy - `Scheduling_Protocol => (POSIX_1003_HIGHEST_PRIORITY_FIRST_PROTOCOL)`

The SatComp process is hosted on a processor CPU. The Processor has a memory limit of 128 MB, it is coded as below.

```
memory RAM Memory_Size => 128000000 bytes;
```

Furthermore, in AADL we have the ability to bind / connect elements together.

- The threads are bound to the process.
- The process and the processor are bound to the system
- And the Memory is bound to the process

```
process SatComp - subcomponents
        : thread TScheduler.impl;
        : thread TDialog.impl;
        : thread Tacquire.impl;
        : thread Tactuator.impl;
        : thread Distributor.impl;

system s - subcomponents
        soft : process SatComp.impl;
        hard : processor CPU;
        RAM : MEMORY RAM;

properties
        Actual_Processor_Binding => (reference(hard)) applies to soft;
        Actual_Memory_Binding => (reference(RAM)) applies to soft;
```

The model is now simulated onto AADL_Inspector tool. The threads are schedulable and the results are as shown below.

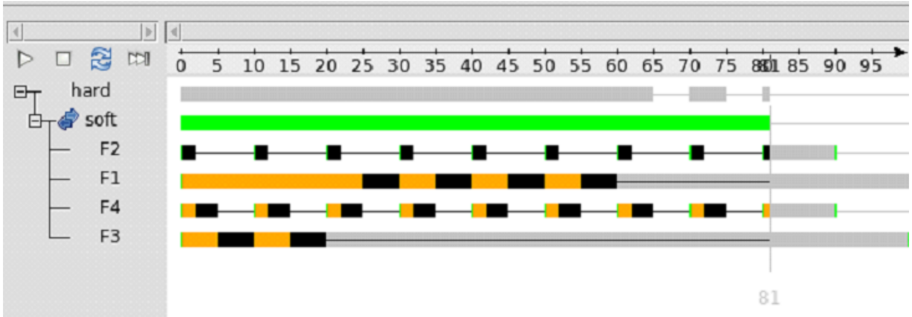



FIGURE 1: INITIAL MODEL SIMULATION

From the Theoretical schedulability test results we can see that the response time for each thread is well within their WCET as stated in the above table

| test | entity | result |
|---|---|---|
| processor utilization factor | hard | Can not apply the feasibility test on processor utilization fa |
| worst case task response time | hard | All task deadlines will be met : the task set is schedulable. |
| response time | hard.soft.F | 4900.00000 |
| response time | hard.soft.F | 1700.00000 |
| response time | hard.soft.F | 500.00000 |
| response time | hard.soft.F | 200.00000 |

FIGURE 2: THEORITICAL SCHEDULIBILITY ANALYSIS

The same results are obtained using the simulation in the aadl tool.

| test | entity | ult |
|---|---|---|
| Task response time computed from simulatio | hard | No deadline missed in the compucessor utilization facto |
| Task response time computed from simulatio | Ethernet | No deadline missed in the compuset is schedulable. |

FIGURE 3: SIMULATION SCHEDULIBILITY ANALYSIS

# 2 Model of the Graphical User Interface Processor

In this part we design a background **thread** `Graphical` that is hosted on different **Process** `GUI`. The GUI Process is run on another **Processor** `CPU_GUI`. The Graphical thread sends the user request command to the TDialog thread of the Satcomp process for further processing. Additionally any information that has to be displayed on the Graphical User Interface is sent by TDialog thread of Satcomp process to the Graphical thread of GUI process.

To facilitate communication between the 2 processors we use TCP/IP bus communication. Also the two threads are designed to have input output ports for establishing this communication. The Thread properties of Graphical thread are as the same as of TDialog thread (both threads are used to communicate user request).

```
thread Graphical
       Compute_Execution_Time => 2000 us .. 2000 us;
       Period => 100000 us;
       Deadline => 100000 us;
       Priority => 1;
       Dispatch_Protocol => Periodic;
```

```
// P_comm is an Input output port of data communication with CPU processor

Process GUI
        : out data port P_comm;
        : in data port P_comm;

        : port  F1.P_comm_out -> TCP_IP_out;
        : port  TCP_IP -> F1.P_comm;

…
Processor CPU_GUI: requires bus access TCP_IP;
…
Processor CPU  : requires bus access TCP_IP;
…
System
        : process GUI.impl;
        : processor CPU_GUI;
        : BUS TCP_IP;

        : PORT soft.TCP_IP_out -> gui.TCP_IP;
        : PORT gui.TCP_IP_out -> soft.TCP_IP;

        Actual_Processor_Binding => (reference(hard_gui)) applies to gui;
        Actual_Connection_Binding => (reference(TCP_IP)) applies to C22;
        Actual_Connection_Binding => (reference(TCP_IP)) applies to C23;
…..
```

Further the hardware components like sensors are added to the model. The sensor data is read by threads TAquire, TActuator and TDialog. The computed wheel command is sent from TActuator to hardware device.

To facilitate this we add data ports and establish connections to communicate.

```
data Sensors_Data
data WheelCommand
data P_comm

thread TDialog
                  : in data port P_comm;
                  : out data port P_comm;
                  : in data port Sensors_Data;

thread Graphical
                  : in data port P_comm;
                  : out data port P_comm;

DEVICE Sensor
                  : out data port Sensors_Data;
                  : requires bus access I2C;

System
         : DEVICE Sensors;
         : BUS ACCESS I2C -> Sensors.I2C;
         : PORT SENSORS.Sensor_Data -> soft.Sensors_inf;
```

With the above setup we have simulated the schedulability test on AADL and the results are as shown below.

| test | entity | result |
|---|---|---|
| processor utilization factor | hard | Can not apply the feasibility test on processor utilizatio |
| worst case task response time | hard | All task deadlines will be met : the task set is schedula |
| response time | hard.soft.F1 | 60.00000 |
| response time | hard.soft.F3 | 20.00000 |
| response time | hard.soft.F4 | 5.00000 |
| response time | hard.soft.F2 | 2.00000 |
| processor utilization factor | hard_gui | Can not apply the feasibility test on processor utilizatio |
| worst case task response time | hard_gui | All task deadlines will be met : the task set is schedula |
| response time | hard_gui.gui.F1 | 20.00000 |
| processor utilization factor | TCP_IP | Can not apply the feasibility test on processor utilizatio |
| worst case task response time | TCP_IP | All task deadlines will be met : the task set is schedula |
| response time | TCP_IP.default.C2 | 1.00000 |

FIGURE 5: THEORITICAL SCHEDULIBILITY ANALYSIS

The above figure shows the theoretical schedulability analysis of the same model. Note the time periods for each thread are scaled downward to enable a successful simulation for observation.

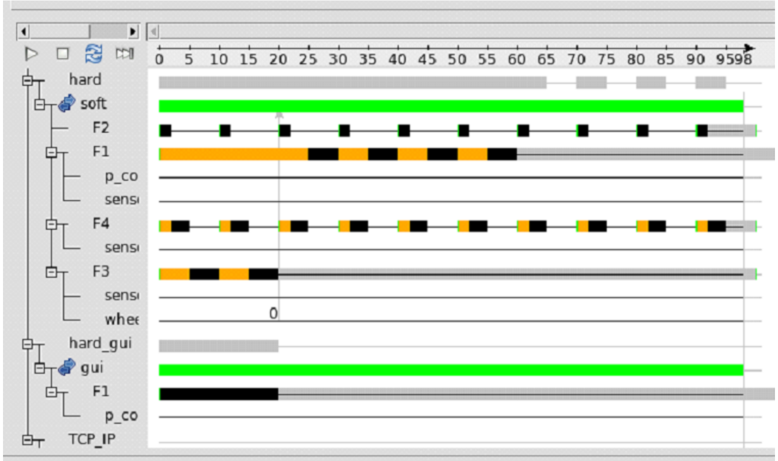


FIGURE 4: MODEL SIMULATION

From the figures we can deduce all task are schedulable and data is sent and received among threads.

# 3 Model of Devices Connected to the System

We modeled the Sensors, Motor, A/D (+ multiplexer) and D/A converter as following:

To model the system to be more realistic we have 2 sets of data ports for each device representing analog and digital and input and output of the data. These data ports are accordingly assigned to the devices

```
DEVICE AD_mult
        : in data port Sensors_Data;
        : out data port Sensors_Data_D;
        : requires bus access I2C;

DEVICE DA_mult
        : in data port WheelCommand;
        : out data port WheelCommand_A;
        : requires bus access I2C;

DEVICE Motor
        : in data port WheelCommand_A;
        : requires bus access I2C;
```

In the system code we add the devices, data ports communicating with the devices and connect the I2C bus

```
system
        : DEVICE Sensors;
        : DEVICE AD_mult;
        : DEVICE DA_mult;
        : DEVICE Motor;
```

These elements are connected to the I2C bus as below:

```
        : BUS ACCESS I2C -> AD_mult.I2C;
        : BUS ACCESS I2C -> DA_mult.I2C;
        : BUS ACCESS I2C -> Motor.I2C;
```

To model the link between computations (SatComp process) and the sensors/actuators, we connected it as following

```
        : PORT SENSORS.Sensor_Data -> AD_mult.Sensor_Data;
        : PORT AD_mult.Sensor_Data_D -> soft.Sensors_inf;
        : PORT soft.WheelCmd -> DA_mult.WheelCommand;
        : PORT DA_mult.WheelCommand_A -> Motor.WheelCommand_A;
```

Here soft is our Satcomp process. We have ports from our Satcomp process connected to the A/D and D/A converter. We have also modelled the fact that the A/D converter gets its data from the sensors then sends it as Digital data to the Satcomp process. In the same way, the D/A converter gets digital data (WheelCmd) from the Satcomp process, then sends the command as an analog signal to the Motor.

These device ports are now connected to the SatComp process

```
process SatComp.impl

       : port  TCP_IP -> F5.P_comm;
       : port  TCP_IP -> F1.P_comm;
       : port F1.P_comm_out -> TCP_IP_out;
       : port F3.WheelCommand -> WheelCmd;
       : port Sensors_inf -> F3.Sensors_Data_D;
       : port Sensors_inf -> F1.Sensors_Data_D;
       : port Sensors_inf -> F4.Sensors_Data_D;
```

# 4 Distributer Thread Model and Resource Refining

Until this model the size of the data to be sent was not considered, here we will specify the data size and the Data_Rate for transmission of this data over the I2C bus.

We consider the following types of Data which are exchanged:

```
data Sensors_Data :: Data_Size => 10 Bytes;
data Sensors_Data_D :: Data_Size => 10 Bytes;
data WheelCommand :: Data_Size => 100 Bytes;
data WheelCommand_A :: Data_Size => 100 Bytes;
data P_comm :: Data_Size => 10 Bytes;
```

The Ethernet interface is assigned a maximum Data rate of 40 Mbits/s

```
bus I2C.impl :: Data_Rate => 40000000 bitsps;
bus TCP_IP.impl :: Data_Rate => 40000000 bitsps;
```

Now a New Thread is added in the design that shall only read data from / to the devices and send the data to the respective threads in the SatComp processor. The modified thread model is as shown below.

| Name | Dispatch_Protocol | Period | Compute_Execution_Time | Deadline | Actual Processor(s) | Priority |
|---|---|---|---|---|---|---|
| soft.F2 | periodic | 1000us | 200us..200us | 1000us | hard | 5 |
| soft.F1 | periodic | 100000us | 2000us..2000us | 100000us | hard | 2 |
| soft.F4 | periodic | 2000us | 300us..300us | 2000us | hard | 4 |
| soft.F3 | periodic | 10000us | 1000us..1000us | 10000us | hard | 3 |
| soft.F5 | periodic | 10000us | 1000us..1000us | 10000us | hard | 1 |
| gui.F1 | periodic | 100000us | 2000us..2000us | 100000us | hard_gui | 1 |

FIGURE 5: FINAL THREAD DISTRIBUTION

The data collected and by the distributer thread is as shown below.

```
thread Distributor
       : out data port P_comm;
       : in data port P_comm;
       : out data port Sensors_Data_D;
       : in data port Sensors_Data_D;
       : out data port WheelCommand;
       : in data port WheelCommand;
```

Finally our system is composed as below

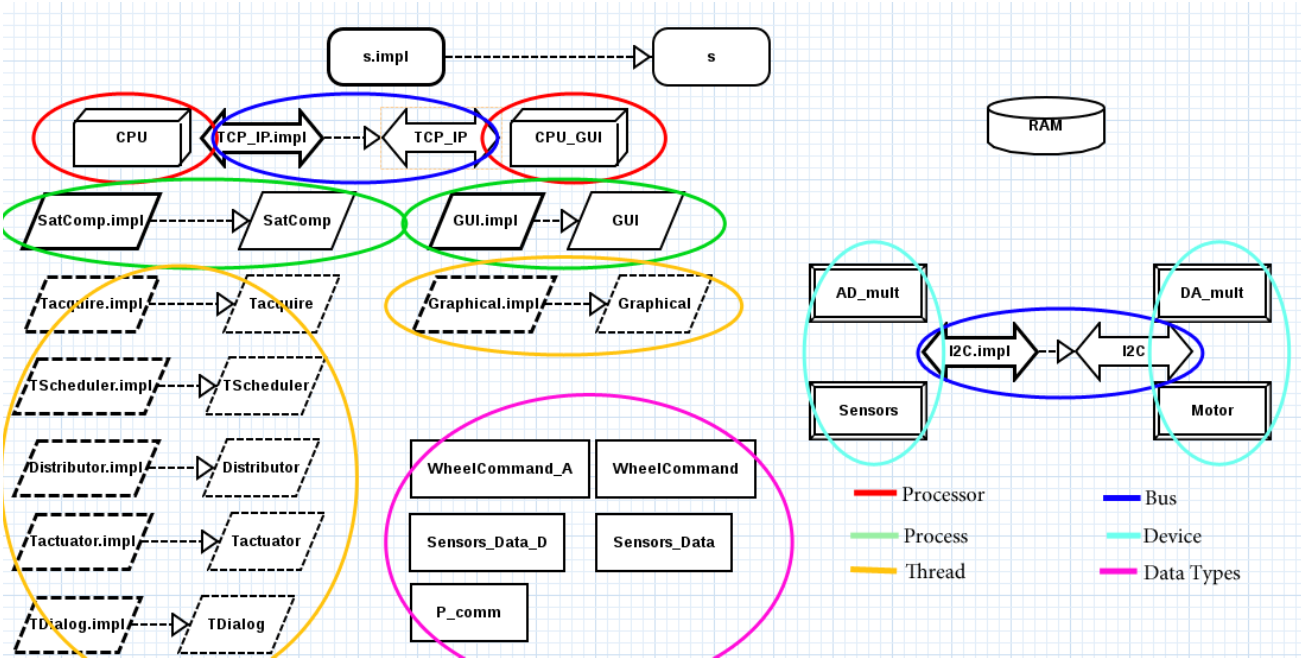



FIGURE 6: GRAPHICAL VIEW

# 5 Simulation Results and Comparison

We obtain with osate the following simulation for 4 threads model (without the distributer thread)

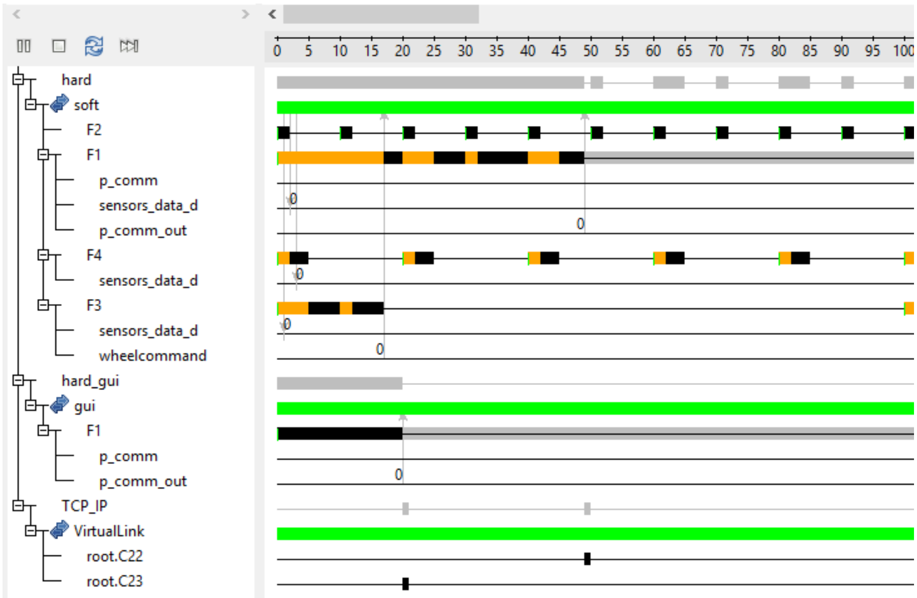
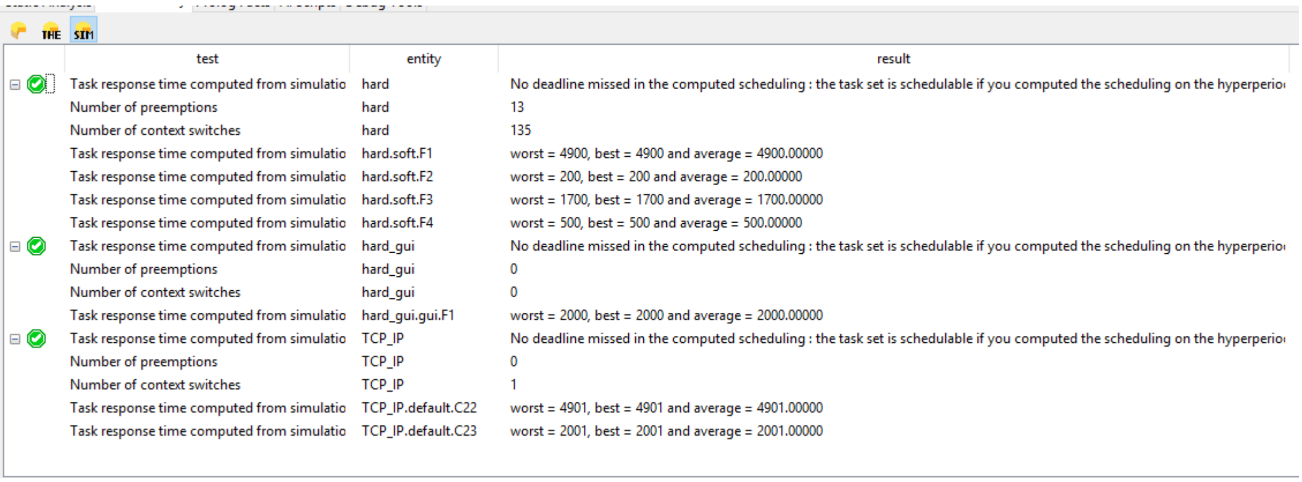
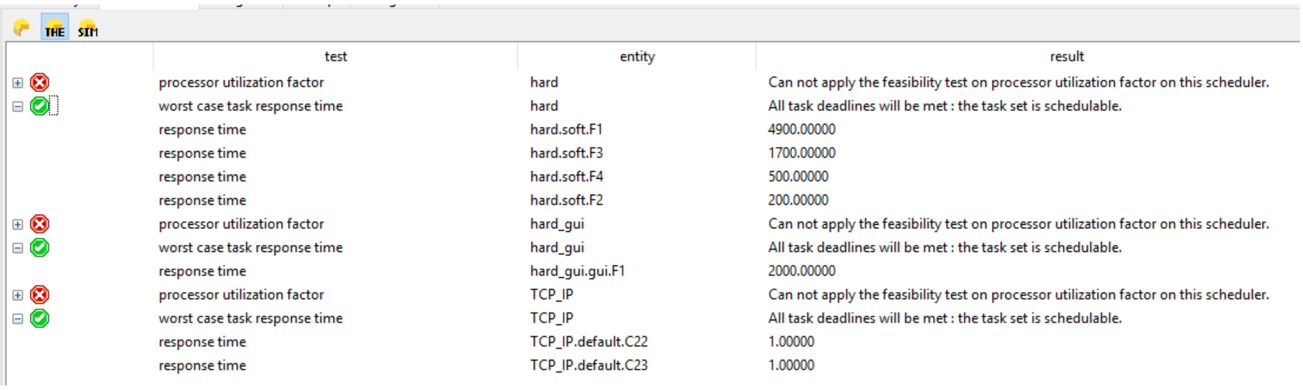



FIGURE 7: SIMULATION OF MODEL WITH 4 THREADS



| test | entity | result |
|---|---|---|
| Task response time computed from simulatio | hard | No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the hyperperio |
| Number of preemptions | hard | 13 |
| Number of context switches | hard | 135 |
| Task response time computed from simulatio | hard.soft.F1 | worst = 4900, best = 4900 and average = 4900.00000 |
| Task response time computed from simulatio | hard.soft.F2 | worst = 200, best = 200 and average = 200.00000 |
| Task response time computed from simulatio | hard.soft.F3 | worst = 1700, best = 1700 and average = 1700.00000 |
| Task response time computed from simulatio | hard.soft.F4 | worst = 500, best = 500 and average = 500.00000 |
| Task response time computed from simulatio | hard_gui | No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the hyperperio |
| Number of preemptions | hard_gui | 0 |
| Number of context switches | hard_gui | 0 |
| Task response time computed from simulatio | hard_gui.gui.F1 | worst = 2000, best = 2000 and average = 2000.00000 |
| Task response time computed from simulatio | TCP_IP | No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the hyperperio |
| Number of preemptions | TCP_IP | 0 |
| Number of context switches | TCP_IP | 1 |
| Task response time computed from simulatio | TCP_IP.default.C22 | worst = 4901, best = 4901 and average = 4901.00000 |
| Task response time computed from simulatio | TCP_IP.default.C23 | worst = 2001, best = 2001 and average = 2001.00000 |

FIGURE 8: SIMULATION SCHEDULIBILITY ANALYSIS – 4 THREAD MODEL



| test | entity | result |
|---|---|---|
| processor utilization factor | hard | Can not apply the feasibility test on processor utilization factor on this scheduler. |
| worst case task response time | hard | All task deadlines will be met : the task set is schedulable. |
| response time | hard.soft.F1 | 4900.00000 |
| response time | hard.soft.F3 | 1700.00000 |
| response time | hard.soft.F4 | 500.00000 |
| response time | hard.soft.F2 | 200.00000 |
| processor utilization factor | hard_gui | Can not apply the feasibility test on processor utilization factor on this scheduler. |
| worst case task response time | hard_gui | All task deadlines will be met : the task set is schedulable. |
| response time | hard_gui.gui.F1 | 2000.00000 |
| processor utilization factor | TCP_IP | Can not apply the feasibility test on processor utilization factor on this scheduler. |
| worst case task response time | TCP_IP | All task deadlines will be met : the task set is schedulable. |
| response time | TCP_IP.default.C22 | 1.00000 |
| response time | TCP_IP.default.C23 | 1.00000 |

FIGURE 9: THEORITICAL SCHEDULIBILITY ANALYSIS – 4 THREAD MODEL

In order to add a fifth thread Tdistributor which is responsible for emitting/receiving packets from the computation code, we compute first the processor utilization time with our 4 threads:

We have then: $\frac{2}{100}+\frac{0,2}{1}+\frac{1}{10}+\frac{0,3}{2}=\frac{2}{100}+\frac{20}{100}+\frac{10}{100}+\frac{15}{100}=\frac{47}{100}$

We observe that the threads used 47% of the processor's capacity. Since we are using a fixed priority scheduling policy and we want to introduce a new low priority thread. This thread has to satisfy the following condition:

$\frac{47}{100}+\mathbf{X}\leq\frac{69}{100}$ where X is the percentage of processors utilization by the thread introduced.

So the maximum processor utilization we can assign to the **Tdistributor** is of **22%**, for our case we decided that it will use **20%**. It has a period and deadline of **100ms** with an execution time of **20 ms**.

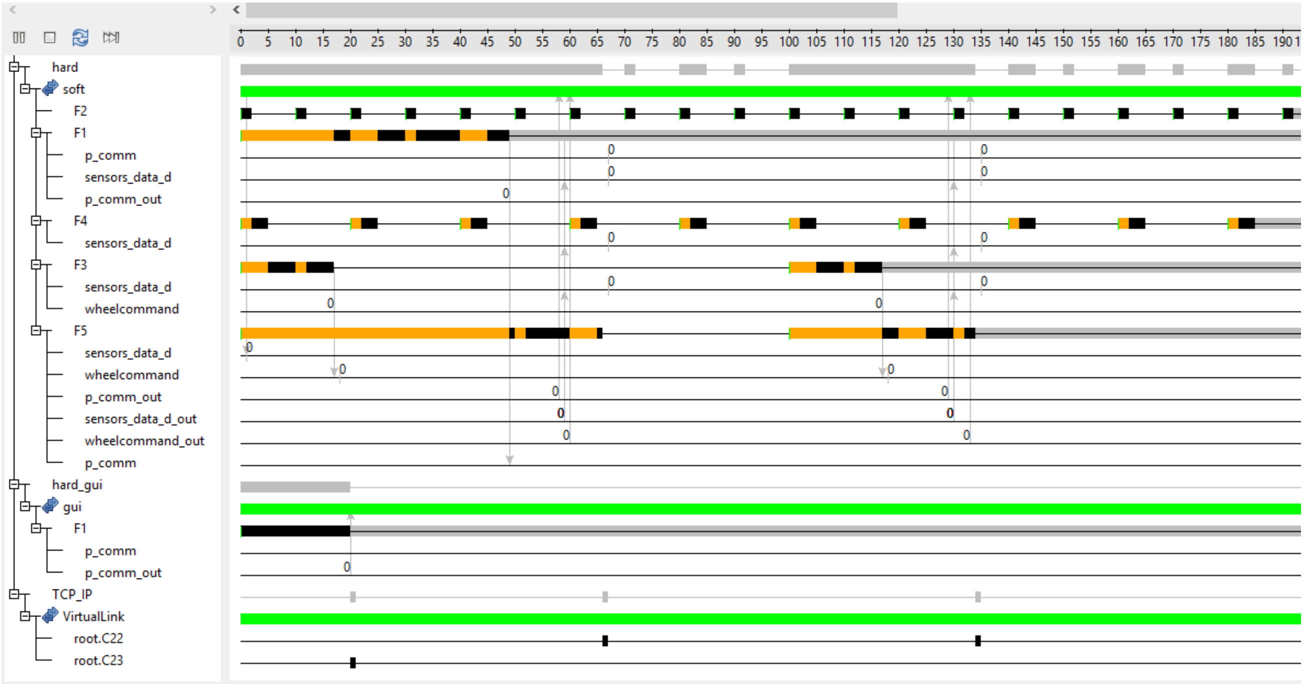



FIGURE 11: SIMULATION OF MODEL WITH 5 THREADS

The analysis shows it is still schedulable

| test | entity | result |
|---|---|---|
| processor utilization factor | hard | Can not apply the feasibility test on processor utilization factor on this scheduler. |
| worst case task response time | hard | All task deadlines will be met : the task set is schedulable. |
| response time | hard.soft.F5 | 6600.00000 |
| response time | hard.soft.F1 | 4900.00000 |
| response time | hard.soft.F3 | 1700.00000 |
| response time | hard.soft.F4 | 500.00000 |
| response time | hard.soft.F2 | 200.00000 |
| processor utilization factor | hard_gui | Can not apply the feasibility test on processor utilization factor on this scheduler. |
| worst case task response time | hard_gui | All task deadlines will be met : the task set is schedulable. |
| response time | hard_gui.gui.F1 | 2000.00000 |
| processor utilization factor | TCP_IP | Can not apply the feasibility test on processor utilization factor on this scheduler. |
| worst case task response time | TCP_IP | All task deadlines will be met : the task set is schedulable. |
| response time | TCP_IP.default.C22 | 1.00000 |
| response time | TCP_IP.default.C23 | 1.00000 |

FIGURE 12: THEORITICAL SCHEDULIBILITY ANALYSIS – 5 THREAD MODEL

Simulation schedulability test

| test | entity | result |
|---|---|---|
| Task response time computed from simulation | hard | No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the hyperperiod. |
| Number of preemptions | hard | 33 |
| Number of context switches | hard | 175 |
| Task response time computed from simulation | hard.soft.F1 | worst = 4900, best = 4900 and average = 4900.00000 |
| Task response time computed from simulation | hard.soft.F2 | worst = 200, best = 200 and average = 200.00000 |
| Task response time computed from simulation | hard.soft.F3 | worst = 1700, best = 1700 and average = 1700.00000 |
| Task response time computed from simulation | hard.soft.F4 | worst = 500, best = 500 and average = 500.00000 |
| Task response time computed from simulation | hard.soft.F5 | worst = 6600, best = 3400 and average = 3720.00000 |
| Task response time computed from simulation | hard_gui | No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the hyperperiod. |
| Number of preemptions | hard_gui | 0 |
| Number of context switches | hard_gui | 0 |
| Task response time computed from simulation | hard_gui.gui.F1 | worst = 2000, best = 2000 and average = 2000.00000 |
| Task response time computed from simulation | TCP_IP | No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the hyperperiod. |
| Number of preemptions | TCP_IP | 0 |
| Number of context switches | TCP_IP | 1 |
| Task response time computed from simulation | TCP_IP.default.C22 | worst = 6601, best = 3401 and average = 3721.00000 |
| Task response time computed from simulation | TCP_IP.default.C23 | worst = 2001, best = 2001 and average = 2001.00000 |

FIGURE 13: SIMULATION SCHEDULIBILITY ANALYSIS – 5 THREAD MODEL

**Worst case response time comparison between task in 5-Thread Model vs 4-Thread Model:**

| Task | 4-Thread Model | 5-Threads Model |
|---|---|---|
| F1 | 4900 | 4900 |
| F2 | 200 | 200 |
| F3 | 1700 | 1700 |
| F4 | 500 | 500 |
| F5 | - | 6600 |
| TCP_IP C22 | 4901 | 3721 |
| TCP_IP C23 | 2001 | 2001 |

All the values given in the table are in micro seconds.

C22: connection from the SatCom to Gui (TCP_IP)
C23: connection from the Gui to the SatCom (TCP_IP)

From the above table we see that the worst response time of the C22 link is higher with 5 threads (6601) compared to with 4 threads (4901). This worst case might happen when F1 sends data (P_comm) for the Gui but F5 has just been executed, so it will have to wait for the next activation of F5. However we know that the worst case response time is a more pessimistic view of the system and as we see that having the $5^{th}$ thread improves (3721instead of 4901) the average response time of the connection (TCP_IP) from the SatComp to the Gui process.

Hence with the above understanding we deduce that the last model with 5 threads seems to be an efficient way for task distribution for this system.

# 6 Conclusion

This project is a good opportunity to apply the advices and the lessons taught. The session gave us a unique exposure to realize a dynamic model and provided us with an insight on safety assessments of dynamic systems.