# REAL TIME SYSTEMS ASSIGNMENT: SCHEDULING FOR AN AUTONOMOUS ROBOT NAVIGATION ALGORITHM

# 30 SEPTEMBER 2022

**Francesco Cerri**
francesco.cerri2@studio.unibo.it
github.com/gnoccoalpesto

## 1 Introduction to the problem

I decided to present a system where a navigation stack for a mobile robot is scheduled. The system presents 4 tasks: highest priority one's purpose is to verify the consistency of the system, whilst others' is to properly let the robot sense the environment and move inside of it.

### 1.1 Task set

Task set is composed by:

1. **tMonitor task** ($\tau_M$): is responsable to checks if all other tasks are correctly operating, respawning them if needed. Has maximum priority to ensure the safety of the system, phased activation since it must wait for other tasks to start, low computation time due to its ping-like behaviour, high period and does not use any resource;
2. **tCamera task** ($\tau_V$): acquires the video from the hardware sensor and publishes it, after preprocessing (for example identifying the AR markers usefull to locate the bodies in the field of view). Has high priority since it's the source of vision for downstream nodes, medium computation time to simulate the presence of a dedicated hardware in the system (i.e. a GPU), medium period and uses (by publishing it) the **Image resource**;
3. **tCollision task** ($\tau_C$): uses processed camera stream to reconstruct the physical objects in the environment. Has lower priority, high computational time, and uses both camera as well as **Objects resource**, the first as input, while it publishes the latter;
4. **tPlanner task** ($\tau_P$): plans robot path using identified objects. Has the lowest priority since its the final user, very long computational time and period and uses as input object resource.

All tasks are **periodic** and have **fixed priority**, with:

$$P(\tau_M) > P(\tau_V) > P(\tau_C) > P(\tau_P)$$

### 1.2 Resources and constraints

Two resources are available in the system:

1. Image resource ($R_I$): the visual data coming from $\tau_V$, used by $\tau_C$;

2. Objects resource ($R_O$): the collision coming from $\tau_C$, used by $\tau_P$.

Both resources have a **single instance**. Scheduling must be done considering that when $\tau_C$ is using the $R_I$ resource, $\tau_V$ must not update it to avoid inconsistency in the data; the same thing must happen for $\tau_P$, $R_O$ and $\tau_C$ respectively. Moreover, **resources are nested** for the same reason.

| Resource | $C(R_k)$ |
|---|---|
| $R_I$ | $P(\tau_V)$ |
| $R_O$ | $P(\tau_C)$ |

Table 1: Resource ceilings based on priorities

### 1.3 Implementation

Simulation of the system is performed using VxWorks7, setting as **clock frequency 20 Hz**. [1] [2] Task set $\Gamma$ presents the following parameters:

| $\Gamma$ | $\Phi_i$ | $C_i$ | $T_i$ | $U_i$ | $t_i^I$ | $\delta_i^I$ | $t_O$ | $\delta_i^O$ |
|---|---|---|---|---|---|---|---|---|
| $\tau_M$ | 20 | 2 | 16 | .125 | | | | |
| $\tau_V$ | 0 | 6 | 30 | .2 | .5 | 5.5 | | |
| $\tau_C$ | 0 | 10 | 40 | .25 | .5 | 9.5 | 3 | 7 |
| $\tau_P$ | 0 | 16 | 60 | .2667 | | | .5 | 7 |

Table 2: Task Set

with an **hyperperiod**H=240 and **harmonic task groups** of periods 16,{30,60},40.

## 2 Analysis

Required conditon of **Processor Utilization Criterion** is satisfied being:

$$\sum_i {}^{C_i}/_{T_i} = .125 + .2 + .25 + .2667 = 0.8417 < 1$$

Since all tasks are periodic, analysis will be performed starting from **Liu-Layland utilization bounds**(LL), switching to **Hyperbolic Bounds**(HB) in case of failure and **Response Time Analysis** (RTA) in last instance.

### 2.1 Blocking times with different Resource Access Protocols

| $\Gamma$ | $B_i^{NPP}$ | $B_i^{HLP}$ | $B_i^{PIP}$ |
|---|---|---|---|
| $\tau_M$ | 9.5 | 0 | 16.5 |
| $\tau_V$ | 9.5 | 9.5 | 9.5 |
| $\tau_C$ | 7 | 7 | 7 |
| $\tau_P$ | 0 | 0 | 0 |

Table 3: Blocking time under different Resource Access Protocols

Computation is performed considering no prehemption for NPP, ceilings for HLP, and (possible) chained blocking for PIP.

[1] Namely, every unit of time represented lasts 50 ms.

[2] This avoids heavy interference by the Windows host system.

### 2.2 Bounds and Response Time Analysis for task set

Considering that

- $B_V^{NPP} == B_V^{HLP} == B_V^{PIP}$
- $B_C^{NPP} == B_C^{HLP} == B_C^{PIP}$
- $B_P^{NPP} == B_P^{HLP} == B_{\tau_P}^{PIP}$

we can use **extended analysis** to confirm feasibility of the task set:

$$\tau_M^{NPP} : U_M + B_M^{NPP} = .125 + .5937 = .718 < LL_1 = 1$$
$$\tau_M^{HLP} : U_M + B_M^{HLP} = .125 + 0 = .125 < LL_1 = 1$$
$$\tau_M^{PIP} : U_M + B_M^{PIP} = .125 + 1.031 = 1.156 > LL_1 = 1$$
$$\tau_V^{NPP} : U_M + U_V + B_V^{NPP} = .125 + .2 + .3167 = .6417 < LL_2 = .828$$
$$\tau_C^{NPP} : U_M + U_V + U_C + B_C^{NPP} = .125 + .2 + .25 + .175 = .75 < LL_3 = .779$$
$$\tau_P^{NPP} : U_M + U_V + U_C + U_P + B_P^{NPP} = .125 + .2 + .25 + .2667 = .8416 > LL_3 = .779$$

**Using bounds, theoretical feasibility of PIP under fixed priorities is automatically not provable**, yet can still be searched for any other protocol. Let's use HB on $\tau_P$:

$$(1 + U_M) \cdot (1 + U_V) \cdot (1 + U_O) \cdot (1 + U_P + B_P^{NPP}) = 1.125 \cdot 1.2 \cdot 1.25 \cdot 1.267 = 2.138 > 2$$

Still feasibility is yet to be proven. RTA is the last (simple) possibility for $\tau_P$:

$$R_P^0 = \sum_i C_k + B_P^{NPP} = 2 + 6 + 10 + 16 = 34 < T_P = 60$$
$$R_P^1 = \sum_j \left\lceil R_P^0 / T_j \right\rceil * C_j + C_P + B_P^{NPP} = 16 + 6 + 12 + 10 = 44 < 60$$
$$R_P^2 = 16 + 6 + 12 + 20 = 54 < 60$$
$$R_P^3 = R_P = 16 + 8 + 12 + 20 = 56 < 60$$

**proving theoretical feasibility under fixed priorities for NPP and HLP**.

## 3 Simulation and results

Using VxWorks I initially show a feasible schedule under **no resource access protocol**. Displayed results are taken from VxW's **System Viewer** tool.

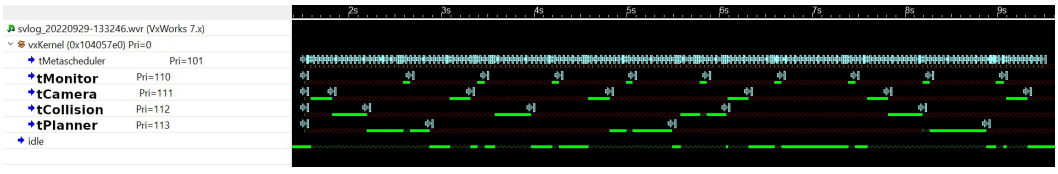



Figure 1: $\Gamma$ under No Protocol and phased activation.

It appears clear that **no tasks register an overrun**, yet response time are pretty long

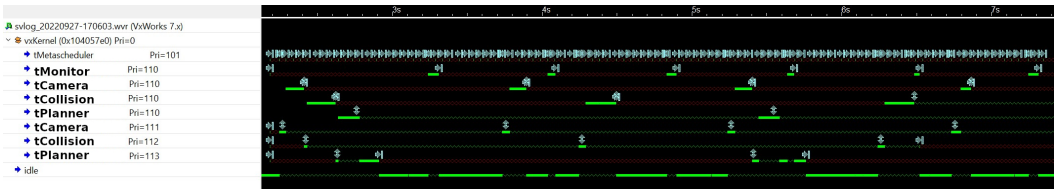
- NOP: $R_P = 2$, $R_V = 7$, $R_C = 15$, $R_P = 29$

suggest the need for a Resource Access Protocol.

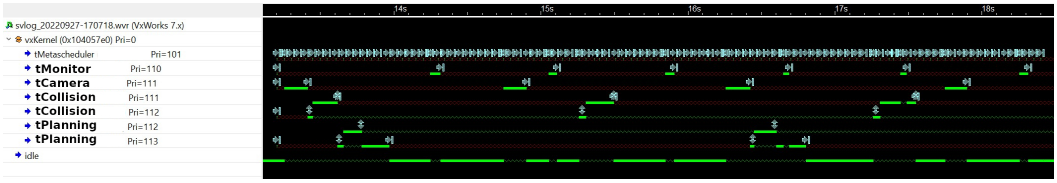
### 3.1 Simulation for R.A.P.s

Here, only simulations for **Non Prehemptive Protocol** (NPP),**Highest Locker Protocol** (HLP) and **Priority Inheritance Protocol**, with **phased activations** are showed.

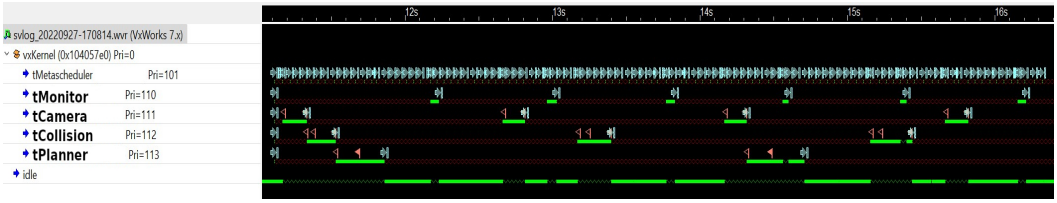
Since it is possible to find a feasible schedule, **feasibilty of PIP under fixed priorities is proven**.




(a)




(b)




(c)

Figure 2: Simulations for phased activations of: (a) NPP, (b) HLP, and (c) PIP protocols

Even in this case **no tasks presents overruns**, yet response times are consistently lower:

- NPP$^{Ph}$: $\mathrm{R}_P = 4, \mathrm{R}_V = 5, \mathrm{R}_C = 9, \mathrm{R}_P = 16$
- HLP$^{Ph}$: $\mathrm{R}_P = 2, \mathrm{R}_V = 5, \mathrm{R}_C = 9, \mathrm{R}_P = 16$
- PIP$^{Ph}$: $\mathrm{R}_P = 2, \mathrm{R}_V = 5, \mathrm{R}_C = 9, \mathrm{R}_P = 15$

For completeness, I also tested **non-phased activations**, resulting in **no overruns** and the following response times:

- NPP$^{N-Ph}$: $\mathrm{R}_P = 2, \mathrm{R}_V = 4, \mathrm{R}_C = 8, \mathrm{R}_P = 15$
- HLP$^{N-Ph}$: $\mathrm{R}_P = 2, \mathrm{R}_V = 4, \mathrm{R}_C = 8, \mathrm{R}_P = 15$
- PIP$^{N-Ph}$: $\mathrm{R}_P = 2, \mathrm{R}_V = 4, \mathrm{R}_C = 8, \mathrm{R}_P = 15$

### 3.2 Conclusions

For what it concerns performances, apart from Phased NPP, all protocol show a small and constant response time for the highest priority tMonitor. Non phased activations also show a smaller response time for all tasks.

Besides that, all R.A.P. behaves very similarly, hence the more logical choice would be selecting the easier one to implement.

### 3.3 Future work

An interesting evolution for this project would be implementing a third resource representing the bandwidth of the communication channel between nodes as a **multi-instance** resource. Other resource will be nested inside of it, to represent the necessity of using the network every time a tasks needs to receive of send data. A possible implementation would be

| $\Gamma$ | $t_B$ | $\delta_i^B$ |
|---|---|---|
| $\tau_M$ | .5 | 1.5 |
| $\tau_V$ | .5 | 5.5 |
| $\tau_C$ | .5 | 9.5 |
| $\tau_P$ | .5 | 7 |

Table 4: Tasks' bandwidth utilization.

## References


[1] Buttazzo, G (2011) Hard Real-Time Computing Systems. Springler.
`https://doi.org/10.1007/978-1-4614-0676-1`

[2] Torroni, G (2021) Lectures of the course "Real Time Systems for Automation, M". Università di Bologna.
`https://www.unibo.it/en/teaching/course-unit-catalogue/course-unit/2021/402384`


## A Code and definitions

For the code and documentation, please refer to
`https://github.com/gnoccoalpesto/vx_rover`