REPORT REAL-TIME ASSIGNMENT A.Y. 2024/2025

Author: SAAD ALI – 0001061473

Objective:

The main objective is to analyse how different resource access protocols handles the task for the following designed task set. The task set can perform and show different results based on the protocol in use.

Task description:

The task set is made assuming a real-time robotics applications, particularly in embedded systems for autonomous or semi-autonomous robots such as robotic arms so here, Task t1 : Sensor Data Acquisition, Task t2: Trajectory Computation, Task t3: Actuator Command Transmission

Task 1: t1 Sensor Data Acquisition

It starts execution after a phase delay of 2 unit. It executes for 1 time unit, then acquires resource Rc and uses it for 3 units. While still holding Rc, it enters a nested critical section by acquiring resource Rb for 1 unit. After that, it completes its execution by running for 1 more unit and then terminates.

Task 2: t2 Trajectory Computation

This task starts after a phase delay of 3 time units. It begins by acquiring resource Ra and uses it for 1 units. Then it continues execution for 1 more unit then acquired Rb for 2 unit. Once the critical section is complete, it executes for 1 more unit and terminates.

Task 3: t3 Actuator Command Transmission

This task begins execution immediately (no phase delay). It begins by running for 1 unit then acquiring resource Rb and uses it for 2 units. Once the critical section is complete, it executes for 1 more unit and terminates.

Task	Phase (ms)	C (ms)	Period = Deadline (ms)	ta	Ra	tb	Rb	tc	Rc
t1	2	5	10	-	-	2	1	1	3
t2	3	6	20	1	1	3	2	-	-
t3	0	4	24	-	-	1	2	-	-

CPU utilization

$$U = \Sigma(C_i / T_i)$$

$$U=6/20 + 4/24 + 5/10 = 0.3 + 0.1667 + 0.5 =0.9667 \approx 96.67\%$$

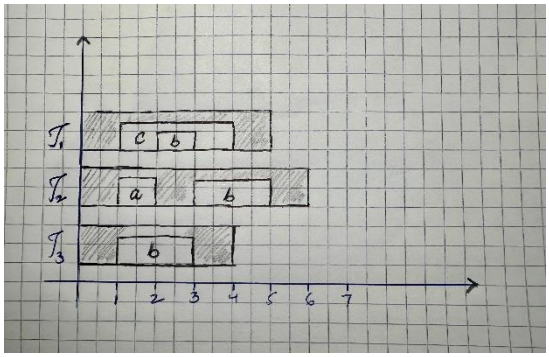
This is within the theoretical bound for Rate Monotonic Scheduling (RMS) with 3 tasks, which is:

$U_{bound}=3(2^{1/3}-1) \approx 0.779$

Feasibility Analysis

t1 (Highest Priority)

Response time:
R1=C1+B1=5+2=7
R1=7≤10=D1 hence its is feasible



t2 (Medium Priority)

R2(0)=C2+B2=6+2=8
R2(1)=C2+B2+⌈R2(0)/T1⌉·C1=6+2+⌈8/10⌉·5=6+2+1·5=
R2(2)=6+2+⌈13/10⌉·5=6+2+2·5=18
R2(3)=6+2+⌈18/10⌉·5=6+2+2·5=18
Fixed point reached.
R2=18≤20=D2 hence it is feasible

t3 (Lowest Priority)

R3(0)=C3+B3=4+0=4
R3(1)=4+⌈4/10⌉·5+⌈4/20⌉·6=4+1·5+1·6=15
R3(2)=4+⌈15/10⌉·5+⌈15/20⌉·6=4+2·5+1·6=20
R3(3)=4+⌈20/10⌉·5+⌈20/20⌉·6=4+2·5+1·6=20
Fixed point reached.
R3=20≤24=D3 hence it is feasible

	NPP		HLP	
Task	**Blocking time**	**Response time**	**Blocking time**	**Response time**
T1	2	7	2	7
T2	2	18	2	18
T3	0	20	0	20

The images attached below shows the gantt chart which is scheduled for the given task set based on NPP and HLP.

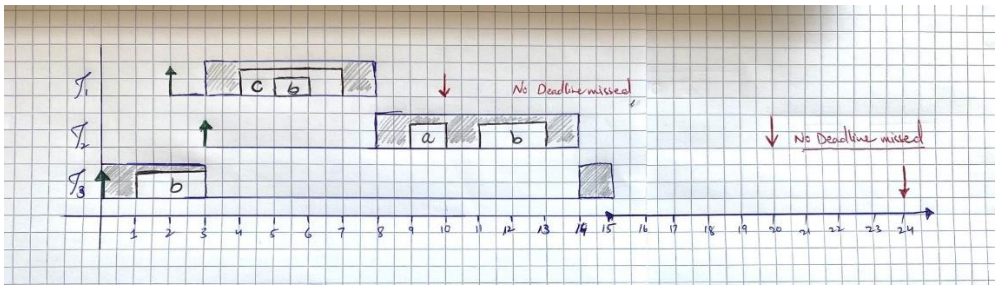


Figure a: NPP Scheduling Timeline (0-15ms) Timeline showing Non-Preemptive Priority scheduling of three tasks (t1, t2, t3) with resource sharing. The schedule demonstrates priority inversion where higher-priority t1 is delayed due to lower-priority task.

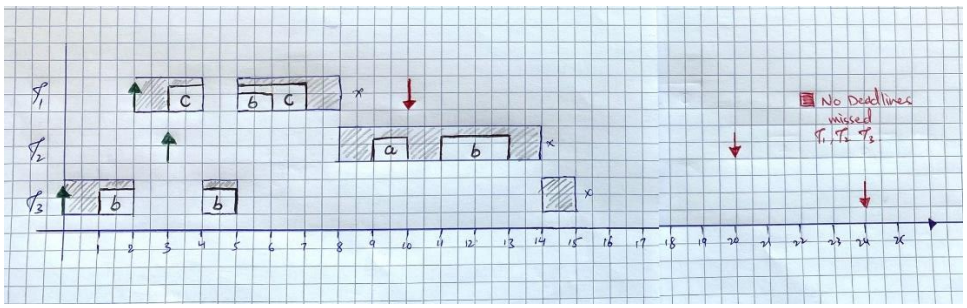


Figure b: HLP Scheduling Timeline (0-15ms) Timeline showing Highest Locker Priority scheduling of the same three tasks with priority inheritance protocol. Compared to NPP, and resource conflicts are resolved through priority inheritance rather than simple blocking. Task t3 inherits higher priority when holding resources needed by t1, preventing priority inversion.

System Viewer Results:

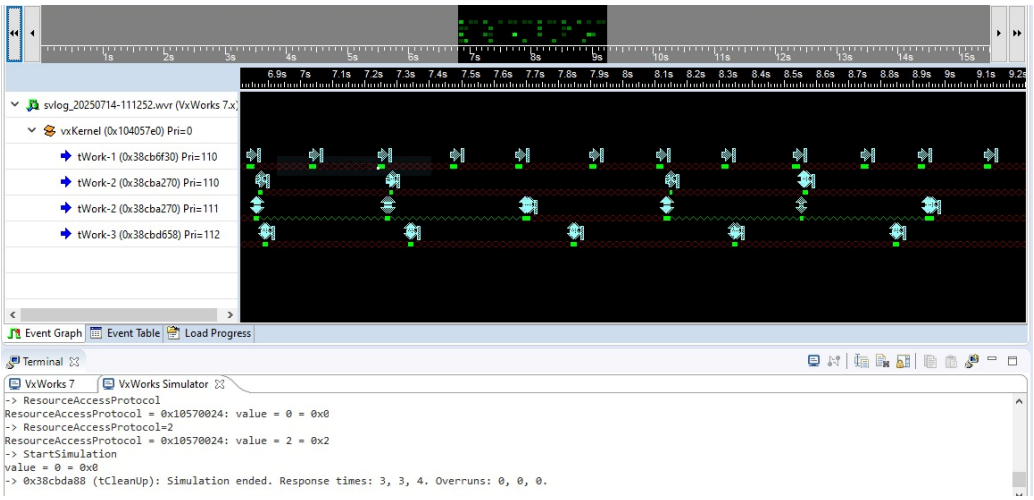


Figure c Event graph for the set of tasks when we have implemented NPP

The VxWorks SystemViewer output confirms that the task set is **schedulable under the Non-Preemptive Protocol (NPP)**. Three tasks tWork-1 (priority 110), tWork-2 (priority 111), and tWork-3 (priority 112) executed with response times of **3 ms**, **3 ms**, and **4 ms** with **zero deadline overruns**.

The green bars indicate active execution, while red segments show tasks waiting due to non-preemptive blocking. The event graph visually validates that higher-priority tasks wait only when a lower-priority task is holding the CPU, consistent with NPP behaviour. No missed deadlines are observed, confirming correct implementation and task-level timing correctness.

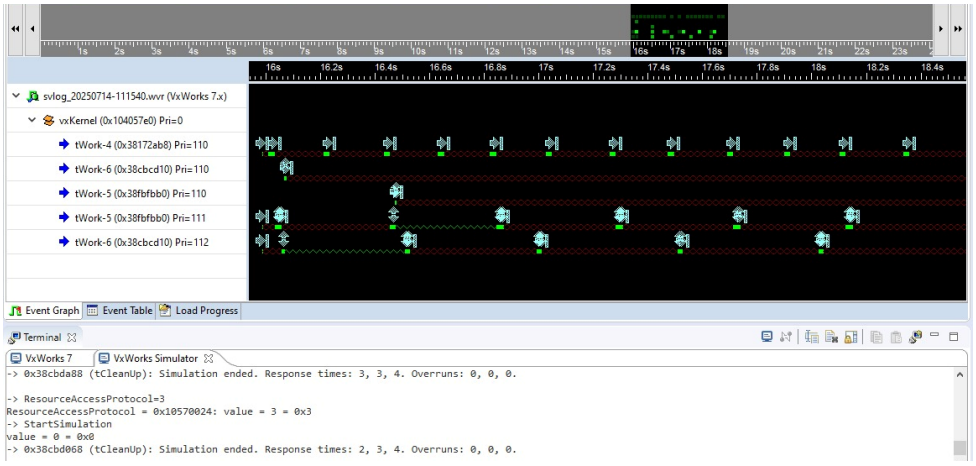


Figure d Event graph for the set of tasks when we have implemented HLP

The VxWorks SystemViewer event graph output confirm that the task set is fully schedulable under the Highest Locker Protocol (HLP). The response times output of 3 tasks are 2 ms, 3 ms, and 4 ms respectively. These values are well within their deadlines of 10 ms, 20 ms, and 24 ms, with zero deadline overruns.

Higher-priority tasks are allowed to preempt lower-priority ones unless a lower-priority task holds a resource accessed by a higher-priority task, which is a key feature of HLP. Compared to the NPP implementation, HLP reduces blocking and maintains efficient CPU utilization by allowing safe preemption outside of critical sections. The observed behaviour aligns with theoretical expectations for HLP, confirming correct protocol implementation and real-time correctness of the system.

NOTE: Modified files application.h and application.c