

Highest Response Ratio Next (HRRN) Scheduling:

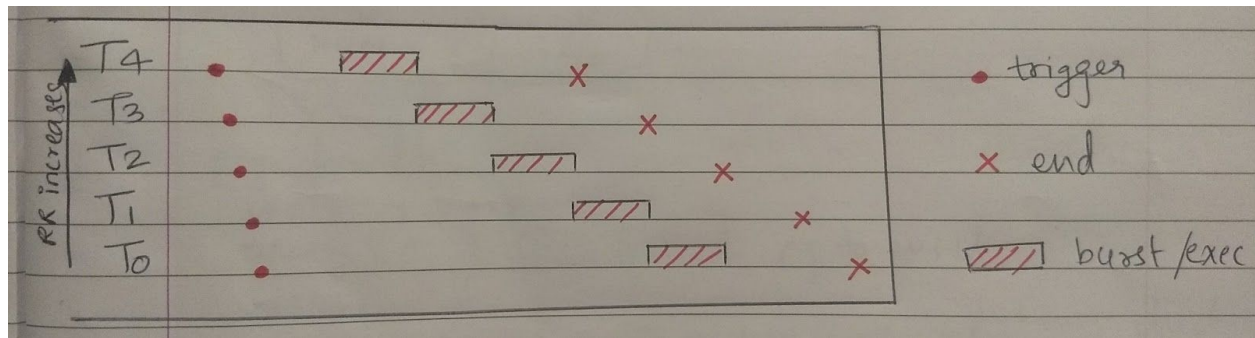
This is a non-preemptive algorithm in which, the scheduling is done on the basis of an extra parameter called Response Ratio. RR is always greater than one.

$$RR = [(Burst\ time + Wait\ time) / Burst\ time] = 1 + (Burst\ time) / (Wait\ time)$$

Example: My test code spawns 5 processes(i=0 to 4) assigning response ratios in increasing order. As seen in the following output, threads with higher response ratios complete their task earlier than threads with comparatively lower response ratios. (For simplicity burst and wait times are pre-decided)

```
for(i=0;i<num_threads;i++) {  
    Calc RR  
    Create pthread tid = i, assign RR.    /* threads created in this order: 0,1,2,3,4  
                                           They exit in this order: 4,3,2,1,0 */  
    Perform task    }
```

```
student@vl69:~$ gcc -w -pthread test.c -lm  
student@vl69:~$ ./a.out  
Main starting  
Spawning Children..  
Thread4 triggered, timestamp=[1573608252.834]  
Response rate of Thread4 = 5  
Thread3 triggered, timestamp=[1573608252.840]  
Response rate of Thread3 = 4  
Thread2 triggered, timestamp=[1573608252.844]  
Response rate of Thread2 = 3  
Thread1 triggered, timestamp=[1573608252.848]  
Response rate of Thread1 = 2  
Thread0 triggered, timestamp=[1573608252.854]  
Response rate of Thread0 = 1  
Thread4 started working, timestamp=[1573608253.570]  
Thread4 finished working, timestamp=[1573608253.606]  
Thread3 started working, timestamp=[1573608253.611]  
Thread3 finished working, timestamp=[1573608253.646]  
Thread2 started working, timestamp=[1573608253.650]  
Thread2 finished working, timestamp=[1573608253.685]  
Thread1 started working, timestamp=[1573608253.687]  
Thread1 finished working, timestamp=[1573608253.722]  
Thread4 exiting, timestamp=[1573608253.725]  
Thread3 exiting, timestamp=[1573608253.731]  
Thread0 started working, timestamp=[1573608253.741]  
Thread0 finished working, timestamp=[1573608253.778]  
Thread2 exiting, timestamp=[1573608253.782]  
Thread1 exiting, timestamp=[1573608253.789]  
Thread0 exiting, timestamp=[1573608257.569]  
Main exiting
```



(X-axis denotes time in above Gantt chart)

Here the order of completion is $T_4 \rightarrow T_3 \rightarrow T_2 \rightarrow T_1 \rightarrow T_0$ in the increasing order of response ratios. Since the algorithm is non-preemptive a thread holds resources until it finishes. Tested on upto 16 processes.