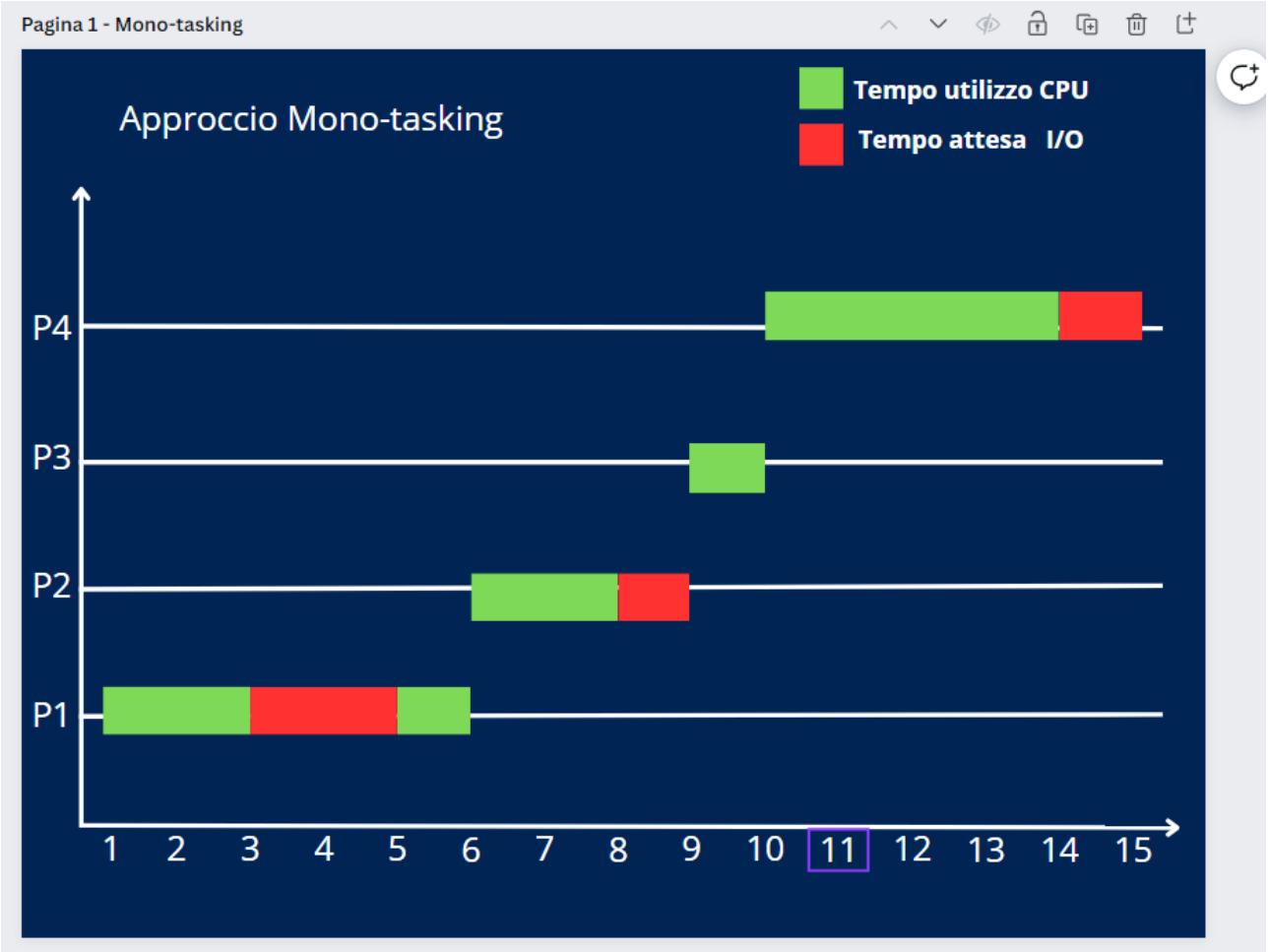
Esercizio S3/L1 (TEAM2)

Oggi vedremo vari approcci con la quale il nostro device può processare le varie informazioni, cambiando il metodo avremmo anche diversi tempi nei quali i vari processi verranno completati.



Mono-Tasking: è un approccio nella quale i processi vengono eseguiti uno alla volta, quindi partendo da P1 fino a P4 verranno eseguiti in ordine:

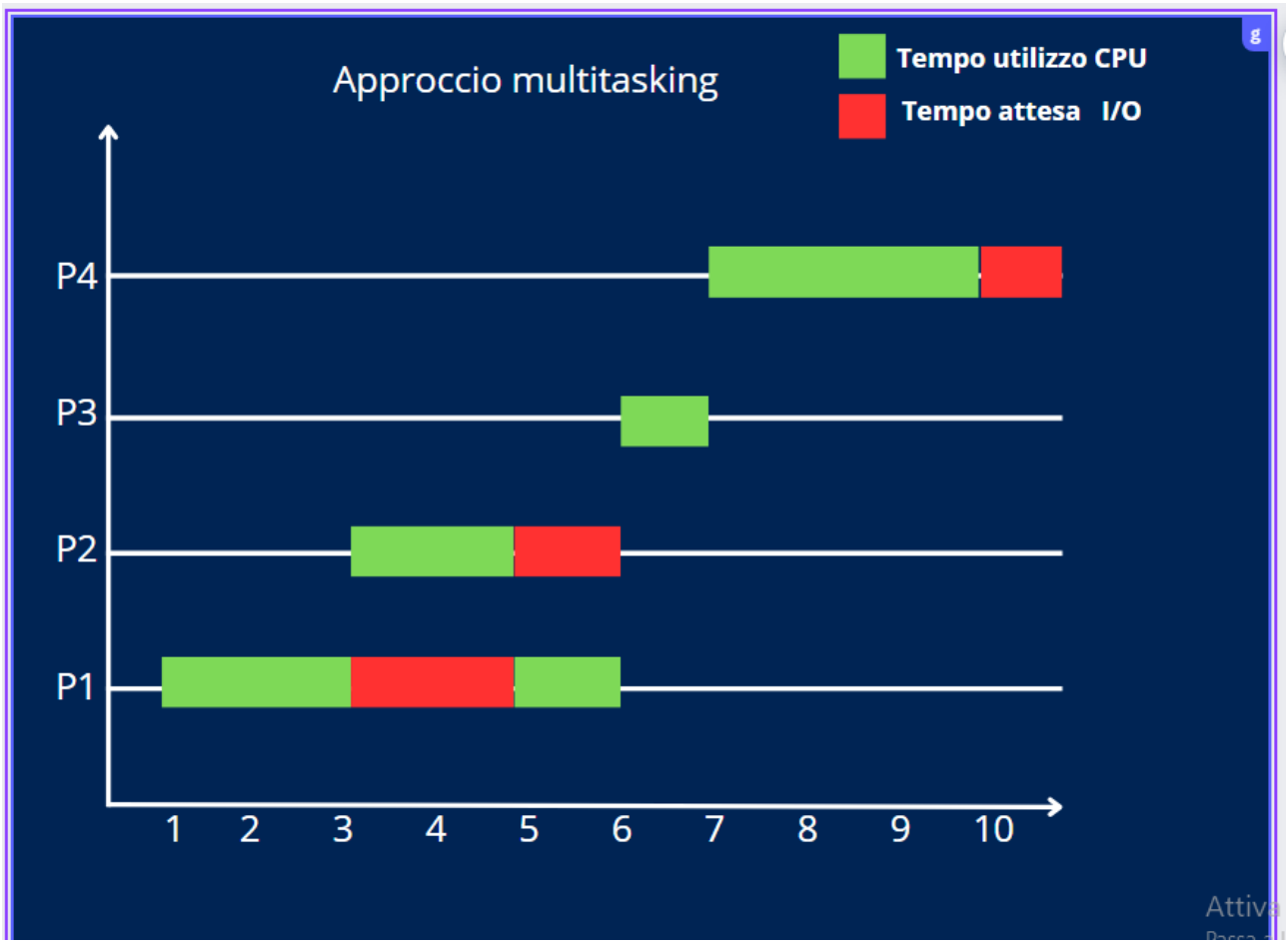
P1: eseguito dopo 6 secondi

P2: eseguito dopo 9 secondi

P3: eseguito dopo 10 secondi

P4: eseguito dopo 15 secondi

Come abbiamo visto nella nostra tabella essi vengono eseguirti senza interruzione partendo dal primo verso l'ultimo aspettando anche il tempo di attesa per ognuno. Purtroppo esso può eseguire solo un processo alla volta, quindi fintanto che P1 sta eseguendo il suo I/O P2 dovrà attendere fino al secondo "7" per iniziare a processare il proprio I/O.
Questo approccio nell'epoca moderna viene usato quasi solo per le macchine industriale. I sistemi Mono-tasking dedicando una significativa frazione del suo tempo ad uno stato di attesa per eventi esterni risulta davvero poco performante rispetto agli standard moderni.



Multi-tasking: Esso rispetto al “mono-tasking” è un approccio che processa più I/O quasi contemporaneamente migliorandone la velocità di esecuzione complessiva. Esso consente di eseguire varie operazioni contemporaneamente, ecco la lista dei tempi di esecuzione:

P1: eseguito dopo 6 secondi

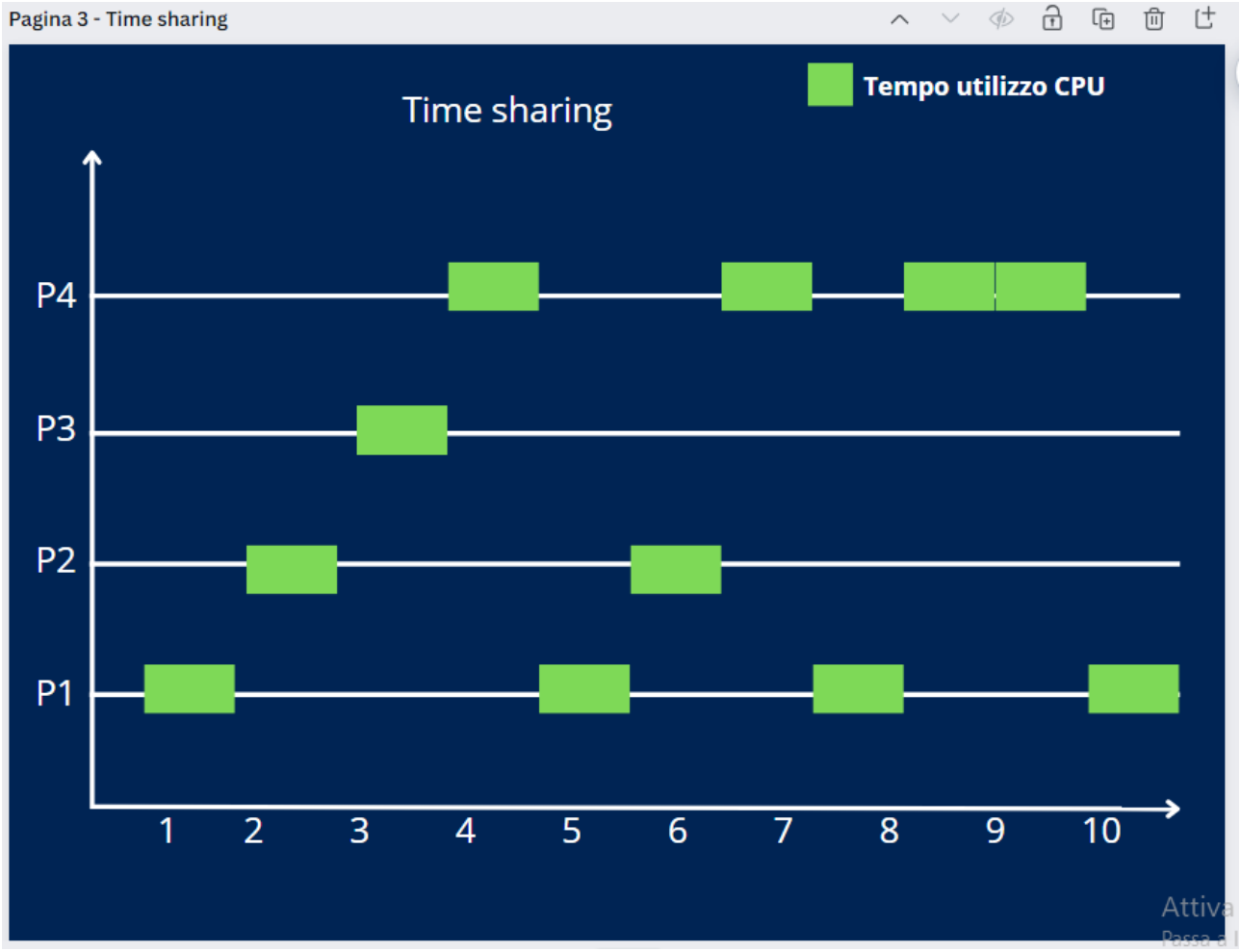
P2: eseguito dopo 6 secondi

P3: eseguito dopo 7 secondi

P4: eseguito dopo 10 secondi

Come possiamo vedere nei tempi di esecuzione dell’approccio se lo confrontiamo con il “mono-tasking” è molto efficiente e rapido diminuendo di 5 secondi il tempo di esecuzione complessivo utilizzando lo stesso numero di processi

Il Multi-tasking viene adoperato principalmente per i “Pc” moderni o device simili per piccole reti. Esso purtroppo gestendo una rete molto tenderà a sovraccaricare il device utilizzato.



Il time-sharing è una tecnica di multitasking in cui il tempo di CPU viene diviso tra i vari processi in modo equo e efficiente. In pratica, ogni processo ottiene una piccola "fetta" di tempo di CPU, prima di essere sospeso e lasciare spazio ad altri processi. Grazie al time-sharing, anche se ci sono molti processi in esecuzione contemporaneamente, il sistema rimane responsivo e interattivo. Ciò significa che gli utenti possono interagire con il sistema e avviare nuove attività senza rallentamenti evidenti.

Il time sharing è una tecnica di multitasking nella quale i tempi di esecuzione vengono suddivisi in base al “Quanto” esso determina la latenza di esecuzione, per il nostro esempio sceglieremo “1 secondo” come durata del “Quanto”:

P1: eseguito dopo 10 secondi

P2: eseguito dopo 6 secondi

P3: eseguito dopo 3 secondi

P4: eseguito dopo 9 secondi

In questo approccio i tempi di attesa sembrano come “sparire” ma in realtà essi sono solo integrati nella latenza da un operazione e un'altra, infatti come possiamo vedere P1 non avrà nessuna attesa anche se ha in altri approcci avrebbe avuto 2 secondi di attesa, un parametro importante da tenere conto invece è il “Tempo di esecuzione dopo attesa” il quale andrà a determinare eventuali “pause” prima di eseguire l’ultimo secondo di un determinato processo; esempio P1 come abbiamo detto in precedenza non ha avuto pause, ma prima di essere eseguito ha comunque saltato un giro di esecuzione finendo così al secondo n10 invece che al n9.

Questo approccio di esecuzione viene impiegato nelle grandi reti dove è necessario che un device riesca a sostenere moltissime operazione contemporaneamente senza collassare.