Planificación de Procesos en Sistemas Operativos: FCFS, SJF, Round–Robin y Prioridad (Estática/Dinámica)

Anonymous ACL submission

Abstract

Evaluamos el impacto de cuatro políticas de planificación (FCFS, SJF, Round–Robin y Prioridad estática/dinámica) sobre rendimiento y equidad en un simulador discreto por *ticks*. Diseñamos escenarios con mezcla de trabajos cortos/largos, perfiles CPU–bound vs. I/O–bound y distintos niveles de concurrencia. Reportamos y discutimos métricas estándar: *turnaround*, espera, respuesta, *throughput* y *fairness*.

1 Introducción

La planificación de CPU determina qué proceso ejecuta y cuándo, influyendo en latencia, utilización y percepción de equidad del sistema. Este trabajo, en el marco del curso de Sistemas Operativos (UNMSM, 2025-2), compara *First-Come, First-Served* (FCFS), *Shortest Job First* (SJF), Round–Robin (RR) y Prioridad (estática y dinámica con envejecimiento), bajo criterios de rendimiento y ausencia de inanición.

En términos clásicos, los planificadores no expropiativos (p.,ej., FCFS y SJF) toman decisiones solo en eventos delimitados (llegada o término de ráfaga), mientras que los expropiativos (p.,ej., RR y prioridad con *preemption*) pueden desalojar al proceso en ejecución para mejorar latencia de respuesta o cumplir políticas de servicio. Cada enfoque materializa un compromiso distinto entre eficiencia (minimizar tiempos promedio y maximizar *throughput*), interactividad (minimizar tiempo de respuesta) y equidad (evitar inanición y sesgos sistemáticos). Fenómenos como el *convoy effect* de FCFS, la sensibilidad de SJF a la estimación de ráfagas o el impacto del *quantum* en RR ilustran que la configuración y la carga determinan el desempeño real del sistema (Eckhardt, 2003; UNS, 2020; ULPGC, 2015; Cabalar, 2012).

Con propósitos didácticos y de reproducibilidad, empleamos un simulador discreto por *ticks* que modela llegadas, ráfagas CPU/I/O, colas de listos y bloqueados, y un costo de cambio de contexto fijo. Este entorno nos permite aislar variables, controlar el grado de concurrencia y construir escenarios sintéticos que representan mezclas de trabajos cortos/largos y perfiles CPU–bound/I/O–bound. Sobre estos escenarios cuantificamos métricas estándar (turnaround, espera, respuesta, *throughput* y un indicador de equidad) y exploramos la sensibilidad a hiperparámetros relevantes: factor de suavizado para SJF, *quantum* para RR y ritmo de envejecimiento en prioridades.

Nuestra hipótesis de trabajo es que: (i) SJF reducirá tiempos promedio en cargas con alta varianza de ráfagas, a costa de penalizar trabajos largos; (ii) RR dominará en latencia de respuesta para cargas interactivas a expensas de mayor sobrecarga si el *quantum* es pequeño; y (iii) la prioridad dinámica mitigará la inanición sin degradar sustancialmente el *throughput* frente a su versión estática. En conjunto, buscamos articular criterios prácticos para elegir políticas según objetivos (interactividad, promedio mínimo o niveles de servicio por clase) y caracterizar los límites de cada algoritmo bajo condiciones controladas.

2 Metodología

2.1 Algoritmos implementados

FCFS (no expropiativo). Atiende en orden de llegada; puede sufrir efecto *convoy* si un trabajo largo bloquea a los demás.

SJF (no expropiativo). Selecciona el trabajo con menor duración estimada. Usamos una estimación exponencial $\hat{b}_{t+1} = \alpha b_t + (1-\alpha)\hat{b}_t$ (por defecto $\alpha = 0.5$).

Round–Robin (expropiativo). Cola FIFO con *quantum* Q; al expirar, se realiza cambio de contexto. Evaluamos $Q \in \{2, 4, 8\}$.

Prioridad Estática (expropiativo). Prela al mayor nivel de prioridad; empates por orden de llegada. Riesgo de inanición en niveles bajos.

Prioridad Dinámica (expropiativo, envejecimiento). Incrementa la prioridad de procesos en espera cada k *ticks* (por defecto $k = 5$) en $+1$, mitigando la inanición.

2.2 Simulador y datos

Simulador discreto por *ticks* con cola de listos, cola de bloqueados por I/O y eventos de llegada/desalojo/desbloqueo. Cada proceso contiene: identificador, tiempo de llegada a_i, traza de ráfagas CPU/I/O y (cuando aplica) prioridad inicial. El costo de cambio de contexto se modela como $c_s = 1$ *tick*.

2.3 Escenarios de prueba

Se definieron tres escenarios; en cada escenario se generaron **5 pruebas** con distintas semillas, **40 procesos** por prueba (llegadas uniformes en una ventana temporal):

- **E1 – Mezcla de duraciones:** 70% cortos (1–5 *ticks*) y 30% largos (20–60), aleatorizados.
- **E2 – CPU vs. I/O:** proporciones variables (50/50, 30/70, 70/30). Los I/O–bound presentan ráfagas de CPU cortas y bloqueos frecuentes; los CPU–bound, ráfagas largas y bloqueos esporádicos.
- **E3 – Concurrencia:** alta (picos de llegada y ráfagas cortas) y baja (llegadas espaciadas, ráfagas más largas).

2.4 Métricas

Para cada proceso i registramos: instante de llegada a_i, primer servicio s_i, finalización f_i, `WaitingTime` (W_i, tiempo en cola de listos reportado por el simulador), `ResponseTime` (R_i), `TurnaroundTime` (T_i).

- **Turnaround** $T_i = f_i - a_i$ (tiempo total en el sistema).
- **Respuesta** $R_i = s_i - a_i$ (latencia hasta la primera CPU).
- **Espera (cola de listos)** W_i (proviene directamente del simulador; no incluye I/O).
- **Throughput** $TP = N/\Delta t$, con N procesos completados en la ventana Δt.

Table 1: E1: mezcla de cortos/largos. Promedios de $\overline{T}$, $\overline{W}$, $\overline{R}$ y TP.

Algoritmo	$\overline{T}$	$\overline{W}$	$\overline{R}$	TP
FCFS	52.69882	48.84233	6.348357	0.2288821/s
SJF	36.46465	32.68845	28.3911	0.2239914/s
RR (Q=4)	53.5065	48.80435	1.269577	0.2212673/s
Prioridad (est.)	37.54732	33.78554	23.2202	0.225072/s
Prioridad (din.)	40.38062	36.35528	25.449	0.2115718/s

Table 2: E2: CPU–bound vs. I/O–bound. Promedios de $\overline{T}$, $\overline{W}$, $\overline{R}$ y TP.

Algoritmo	$\overline{T}$	$\overline{W}$	$\overline{R}$	TP
FCFS	50.13556	45.60741	3.579559	0.2428336/s
SJF	34.94983	30.14719	25.10346	0.2318398/s
RR (Q=4)	52.48986	47.24366	1.1648133	0.2289186/s
Prioridad (est.)	37.93001	33.03173	23.40229	0.2266763/s
Prioridad (din.)	36.10918	31.27518	21.5585	0.2303594/s

3 Resultados y Discusión

Discusión (E1). En la mezcla de cortos/largos, **SJF** obtiene los menores promedios de tiempo de completación y de espera ($\overline{T}$=36.46, $\overline{W}$=32.69), confirmando su ventaja cuando hay trabajos cortos que "limpian" la cola. **RR** entrega la mejor latencia de primera respuesta ($\overline{R}$=1.27) gracias a la expropiación frecuente, pero con $\overline{T}$ y $\overline{W}$ más altos. El *throughput* más alto lo logra **FCFS** (TP=0.2289/s), seguido de **Prioridad (est.)** (0.2251/s); **Prioridad (din.)** queda por detrás en TP (0.2116/s). En suma: si el objetivo son tiempos promedio bajos, SJF; si es latencia inicial, RR; si es productividad, FCFS (con Prioridad est. cerca).

Discusión (E2). Con mezcla CPU/I–O, **SJF** vuelve a liderar en $\overline{T}$/$\overline{W}$ (34.95/30.15). **Prioridad (din.)** se le aproxima (36.11/31.28), señal de que el envejecimiento ayuda a no "congelar" colas bajas. **RR** mantiene la mejor $\overline{R}$ (1.16), coherente con procesos que alternan CPU e I/O, aunque su TP no destaca. El mayor *throughput* lo obtiene **FCFS** (0.2428/s), seguido de **SJF** (0.2318/s). Lectura operativa: para productividad con sobrecarga mínima, FCFS; para promedios bajos, SJF; para interactividad, RR.

Discusión (E3). Bajo alta concurrencia, **SJF** domina tanto en tiempos ($\overline{T}$=29.20, $\overline{W}$=26.54) como en *throughput* (TP=0.3275/s). **RR** conserva la mejor $\overline{R}$ (2.03), lo que beneficia cargas interactivas, pero queda por detrás en $\overline{T}$/$\overline{W}$ y en TP. **FCFS** y **RR** le siguen de cerca en productividad (0.3172 y 0.3165/s); ambas variantes de

Table 3: E3: distintos niveles de concurrencia. Promedios de $\overline{T}$, $\overline{W}$, $\overline{R}$ y TP.

Algoritmo	$\overline{T}$	$\overline{W}$	$\overline{R}$	TP
FCFS	47.05365	44.24879	7.624824	0.3171851/s
SJF	29.20271	26.54047	23.54146	0.327546/s
RR (Q=4)	49.21656	45.91221	2.027393	0.3165274/s
Prioridad (est.)	38.03285	35.12702	23.69838	0.2995332/s
Prioridad (din.)	38.0152	35.14764	23.84064	0.2990064/s

Prioridad quedan intermedias en tiempos y por debajo en TP ($\approx 0.299/\mathrm{s}$).

4 Conclusiones

Los resultados muestran que la elección de la política depende del objetivo operativo y del perfil de carga:

- **Tiempos promedio ($\overline{T}, \overline{W}$). SJF** es consistentemente la mejor política en E1–E3. Si el KPI central es reducir tiempos medios de finalización/espera, SJF es la elección por defecto.
- **Latencia de primera respuesta ($\overline{R}$). RR** domina en los tres escenarios, ofreciendo respuestas iniciales rápidas a costa de peores promedios y, a menudo, menor TP.
- **Productividad (TP). FCFS** logra el mayor TP en E1–E2 (mezcla general y CPU/I–O), mientras que **SJF** pasa al frente en alta concurrencia (E3).
- **Prioridades. Prioridad (din.)** mejora la equidad y reduce esperas extremas respecto a la estática, pero no lidera en promedios ni en TP; su valor diferencial es la mitigación de inanición.
- **Guía rápida de selección.** Interactividad/UX $\Rightarrow$ **RR**; tiempos promedio bajos $\Rightarrow$ **SJF**; throughput con baja sobrecarga en mezclas CPU/I–O $\Rightarrow$ **FCFS**; mucha concurrencia con foco en productividad $\Rightarrow$ **SJF**; equidad/evitar inanición $\Rightarrow$ **Prioridad (din.)**.
- **Sensibilidad a hiperparámetros.** El *quantum* (RR) y el envejecimiento (Prioridad din.) desplazan estos equilibrios: *quantums* más pequeños mejoran $\overline{R}$ pero pueden degradar TP; un envejecimiento agresivo protege colas bajas pero puede alejarse del óptimo de productividad. Deben calibrarse según el perfil real de carga y los niveles de servicio.

En síntesis. Para *interactividad*, **RR**; para *tiempos promedio* bajos, **SJF**; para *throughput* con mezcla CPU/I–O y baja sobrecarga, **FCFS**; con *alta concurrencia* y foco en productividad, **SJF**. Si se prioriza *equidad* y evitar inanición, **Prioridad (din.)**. El *quantum* (RR) y el envejecimiento (prioridad din.) deben calibrarse según el perfil de carga y los niveles de servicio.

Repositorio. Código y experimentos: https://github.com/lightbluedev05/SimulacionSO.git.

Limitaciones

No modelamos multiprocesamiento/NUMA, jerarquías de caché ni *preemption cost* dependiente de la arquitectura; las ráfagas CPU/I/O usan distribuciones simples; la equidad de Jain no captura criterios por clases de servicio.

References

D. Eckhardt. *Scheduling. Lecture 17b, 15-410 Operating Systems*. Carnegie Mellon University, 2003. https://www.cs.cmu.edu/~410-f03/lectures/L17b_Scheduling.pdf

Universidad Nacional del Sur (Departamento de Ciencias e Ingeniería de la Computación). *Sistemas Operativos — Planificación de Procesos*. 2020. https://cs.uns.edu.ar/~so/data/apuntes/SO-2020-mod%2007.pdf

Universidad de Las Palmas de Gran Canaria (FSO). *FSO-02.2: Planificación de procesos*. 2015. https://sopa.dis.ulpgc.es/fso/teoria/pdf/FSO-02.2-Planificacion%20de%20procesos.pdf

P. Cabalar. *Sistemas Operativos — Tema III: Procesos (Planificación)*. Universidade da Coruña, Departamento de Computación, 2012. https://www.dc.fi.udc.es/~so-grado/SO-Procesos-planif.pdf