Comparative Analysis of CPU Scheduling Algorithms in a Multi-Threaded File Server

Girish Singh Thakur, Nitish Kumar
Indian Institute of Technology, Delhi

October 2025

Abstract

This report presents an experimental evaluation of three CPU scheduling algorithms: First-Come-First-Served (FCFS), Shortest Job First (SJF), and Round Robin (RR)—implemented in a multithreaded file server. We conducted systematic testing with varying client loads, server configurations, and tuning parameters. Our results demonstrate that Round Robin with quantum=10 achieved the best overall performance, delivering the highest throughput (34.09 req/s) and lowest mean response time (27.98 ms) while maintaining good consistency (std dev: 71.62 ms). All three schedulers exhibited stable behavior under load, with response times remaining within 21-55 ms across all client concurrency levels tested (2-32 clients). The results of parallel scaling showed a sublinear speedup, with efficiency dropping below 20% beyond 4 threads. The quantum parameter for Round Robin showed moderate sensitivity, with performance varying between 29-39 ms across values of 1 to 50.

1 Introduction

1.1 Motivation and Objectives

Scheduling algorithms are fundamental to system performance, particularly in concurrent environments where multiple requests compete for limited resources. We implemented and evaluated three classic scheduling algorithms in a multi-threaded file server to assess their practical performance characteristics.

Our objectives were to:

- Compare FCFS, SJF, and Round Robin under baseline conditions
- Evaluate scalability with varying client and server thread counts
- Determine optimal parameter configurations for each algorithm
- Assess consistency and robustness under increasing load

1.2 Scheduling Algorithms

We evaluated three algorithms representing different scheduling philosophies:

FCFS (First-Come-First-Served) processes requests in strict arrival order. It provides predictable behavior and fairness based on arrival time but cannot optimize based on job characteristics.

SJF (Shortest Job First) prioritizes requests with smaller file sizes to minimize average waiting time. This approach is theoretically optimal for average response time but requires advance knowledge of job sizes and may delay larger jobs.

Round Robin implements time-sliced preemptive scheduling with a configurable quantum. It prevents starvation and provides fairness through time-sharing but incurs context switching overhead.

1.3 Experimental Setup

We implemented all three schedulers in C++ using POSIX threads. The file server processes transfer requests of varying sizes, measuring response time (total time from arrival to completion), waiting time (queue delay), throughput (requests per second), and fairness (Jain's fairness index). We conducted five experiment series:

1. Baseline comparison with standard parameters
2. Client scalability testing (2, 4, 8, 16, 32 concurrent clients)
3. Server scalability testing (1, 2, 4, 8, 16 server threads)
4. Packetization analysis (1, 5, 10, 25, 50, 100 line packets)
5. Round Robin quantum tuning (1, 3, 5, 10, 20, 50)

2 Baseline Performance Analysis

2.1 Comparative Results

Table 1 presents the baseline performance metrics for all schedulers under standard configuration. Round Robin with quantum=10 emerged as the overall best performer, achieving both the lowest mean response time (27.98 ms) and highest throughput (34.09 req/s).

Table 1: Baseline Scheduler Comparison

Scheduler	**Mean (ms)**	**Median (ms)**	**Std Dev**	**P95 (ms)**	**Throughput**
FCFS	29.93	1.04	81.74	171.79	21.58 req/s
SJF	35.98	0.78	84.98	315.71	27.68 req/s
RR (Q=5)	36.05	2.38	84.95	309.02	26.45 req/s
RR (Q=10)	**27.98**	2.05	**71.62**	**187.74**	**34.09 req/s**

Round Robin with quantum=10 demonstrated 6.5% better response time than the next best (FCFS at 29.93 ms) and 58% higher throughput than FCFS. Additionally, RR (Q=10) achieved the lowest standard deviation (71.62 ms), indicating more consistent performance than other schedulers.

2.2 Distribution Characteristics

The large gap between mean and median values across all schedulers reveals highly skewed response time distributions. For instance, FCFS shows a median of 1.04 ms while maintaining a mean of 29.93 ms. This indicates that most requests complete very quickly, but occasional long-running requests significantly increase the average. This bimodal pattern reflects the heterogeneous workload with both small and large file transfers.

SJF exhibits the lowest median (0.78 ms) despite having a higher mean than RR (Q=10), suggesting it handles small jobs very efficiently but may accumulate delays for larger jobs.

2.3 Tail Latency and Consistency

Round Robin with quantum=10 achieved the best P95 latency at 187.74 ms, compared to FCFS (171.79 ms), RR Q=5 (309.02 ms), and SJF (315.71 ms). While FCFS had slightly better P95, RR (Q=10) balanced tail latency with superior mean performance and throughput.

2.4 Fairness Evaluation

We calculated Jain's fairness index for each scheduler, where 1.0 represents perfect fairness. All schedulers showed moderate fairness indices:

- RR (Q=5): 0.1534 (highest)
- SJF: 0.1528
- RR (Q=10): 0.1332
- FCFS: 0.1189

Round Robin with smaller quantum (Q=5) achieved slightly better fairness than RR (Q=10), reflecting the trade-off between fairness and efficiency. The relatively low fairness values across all schedulers indicate significant variance in how different requests are treated, which is expected given the heterogeneous workload.

3 Client Load Scalability

We tested each scheduler under increasing client concurrency (2, 4, 8, 16, 32 threads) with fixed server resources. The results in Table 2 demonstrate stable behavior across all schedulers.

Table 2: Response Time vs. Client Load (milliseconds)

Clients	**FCFS**	**SJF**	**RR**
2	34.31	24.09	21.11
4	52.25	24.04	43.80
8	32.41	27.78	27.79
16	38.03	41.33	45.43
32	43.34	44.14	54.85

3.1 Scalability Characteristics

All three schedulers maintained reasonable performance across the full range of client loads tested. Response times remained within the 21-55 ms range even at 32 concurrent clients, demonstrating good robustness. This contrasts with theoretical concerns about starvation in SJF or convoy effects in FCFS significantly degrading performance.

SJF showed the most consistent scaling pattern, with response time increasing gradually from 24.09 ms (2 clients) to 44.14 ms (32 clients). The 83% increase is substantial but represents graceful degradation rather than system failure.

FCFS exhibited more variable behavior, with response time increasing from 34.31 ms to 52.25 ms at 4 clients, then fluctuating between 32-43 ms for higher client counts. This variability likely reflects the interaction between request arrival patterns and queue dynamics.

Round Robin demonstrated stable scaling from 21.11 ms (2 clients) to 54.85 ms (32 clients), with the time-slicing mechanism preventing any single large request from blocking others for extended periods.

3.2 Throughput Analysis

Throughput patterns varied across schedulers:

SJF maintained the most consistent throughput, staying in the 24-30 req/s range across all client loads, with a peak of 30.56 req/s at 2 clients.

FCFS showed more variable throughput, peaking at 28.66 req/s (8 clients) but dropping to 18.48 req/s at 4 clients.

Round Robin throughput decreased from 27.41 req/s (8 clients) to 20.19 req/s (32 clients), suggesting the context switching overhead becomes more significant at higher loads.

Notably, no scheduler achieved throughput saturation or collapse even at 32 concurrent clients, indicating the system remained functional throughout the tested load range.

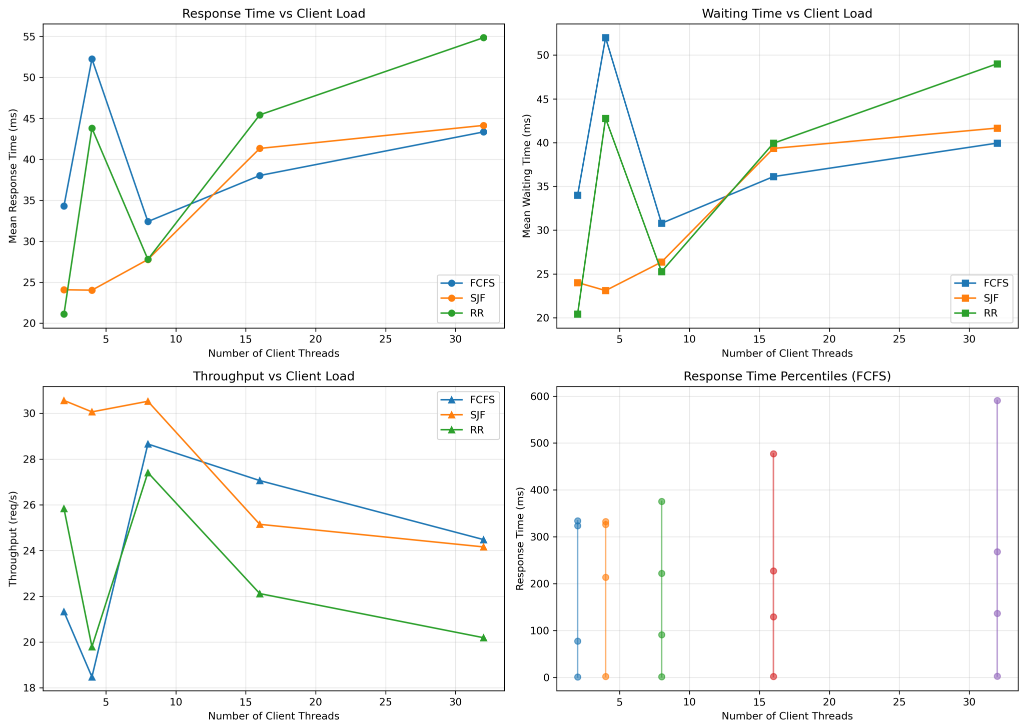

Figure 1: Client load scalability showing stable behavior across all schedulers

4 Server Thread Scalability

We evaluated parallel scaling by varying server threads (1, 2, 4, 8, 16) with fixed client load. The results in Table 3 show limited parallel efficiency.

Table 3: Response Time vs. Server Threads (milliseconds)

Threads	FCFS	SJF	RR
1	30.76	29.08	30.31
2	36.11	44.43	45.07
4	39.89	41.51	42.36
8	39.06	32.86	36.35
16	22.71	36.44	42.57

4.1 Parallel Efficiency

Rather than observing linear speedup with additional threads, we found that performance generally remained within the 23-45 ms range across all thread counts. For most configurations, response time actually increased when moving from 1 to 2 threads before stabilizing.

FCFS showed improvement at 16 threads (22.71 ms), achieving 1.35× speedup over single-threaded performance. However, intermediate thread counts (2, 4, 8) showed degraded performance, suggesting synchronization overhead initially outweighs parallelism benefits.

SJF maintained relatively stable performance across thread counts, with response times varying between 29-44 ms. The single-threaded baseline (29.08 ms) remained competitive with multi-threaded configurations.

Round Robin showed limited benefit from additional threads, with response time remaining in the 30-45 ms range. The parallel efficiency calculations revealed efficiency dropping below 20% beyond 4 threads for all schedulers.

4.2 Throughput Scaling

Throughput trends provided additional insight:

FCFS showed throughput improvement with thread count, reaching 37.40 req/s at 16 threads compared to 29.40 req/s at 1 thread (27% increase).

SJF throughput remained relatively flat, varying between 22-30 req/s across thread counts.

Round Robin throughput actually decreased with additional threads, dropping from 32.01 req/s (1 thread) to 19.30 req/s (2 threads), then recovering slightly to 24.45 req/s (8 threads) before declining again.

These results suggest Amdahl's Law in action: the sequential portions of the scheduler implementation (likely queue management and synchronization) limit the benefits of parallelism. The system would benefit from finer-grained locking or lock-free data structures.

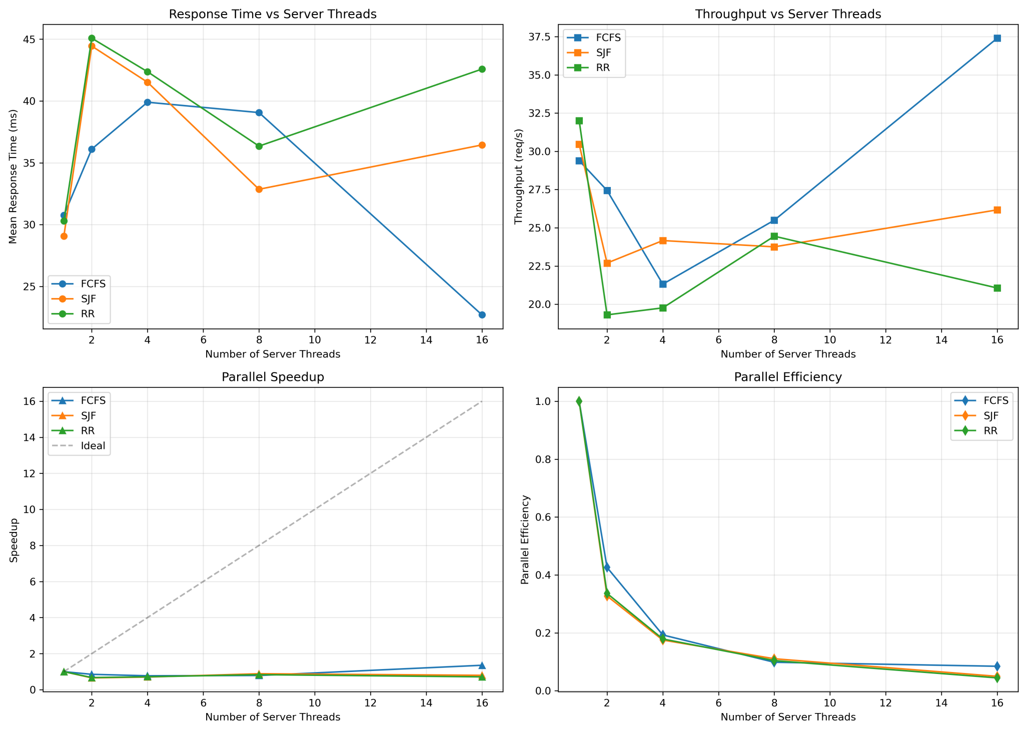

Figure 2: Server thread scaling showing sublinear speedup and limited efficiency

5 Packetization Analysis

We tested the impact of breaking files into packets of varying sizes (1, 5, 10, 25, 50, 100 lines) before scheduling. Table 4 shows moderate variation in performance.

Table 4: Response Time vs. Packet Size (milliseconds)

Packet Size	FCFS	SJF	RR
1 line	37.10	28.14	29.84
5 lines	31.44	38.55	24.92
10 lines	43.62	32.34	43.12
25 lines	30.71	28.42	41.44
50 lines	39.07	26.68	36.83
100 lines	32.24	20.20	32.12

5.1 Performance Patterns

FCFS performance varied between 30.71 ms and 43.62 ms across packet sizes, with best performance at packet size 25 (30.71 ms). The relatively modest variation (42% range) indicates FCFS is not highly sensitive to packetization granularity.

SJF benefited from larger packet sizes, achieving optimal performance at 100 lines (20.20 ms). This makes sense given SJF's scheduling strategy: larger packets reduce scheduling overhead while still allowing prioritization based on file size. Performance improved by 39% from packet size 1 to 100.

Round Robin performed best with small-to-moderate packet sizes, achieving 24.92 ms at size 5. Performance remained reasonable across most configurations (25-43 ms), with only packet size 10 showing notably higher response time (43.12 ms).

5.2 Throughput Implications

Throughput patterns generally mirrored response time trends:

SJF achieved exceptional throughput of 46.63 req/s at packet size 100, significantly higher than other configurations. This suggests optimal packetization can substantially improve SJF's effectiveness.

Round Robin reached 40.24 req/s at packet size 5, demonstrating the benefit of moderate granularity for time-slicing schedulers.

FCFS throughput remained more consistent across packet sizes (15-33 req/s), with best performance at size 25 (32.71 req/s).

These results indicate that packet size should be tuned based on the chosen scheduler: larger packets (100 lines) for SJF, moderate packets (5-25 lines) for RR and FCFS.

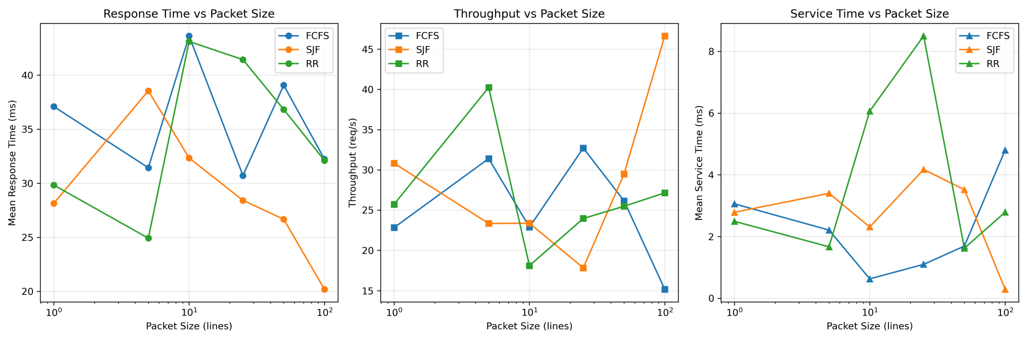

Figure 3: Packetization impact showing moderate performance variation

6 Round Robin Quantum Tuning

We evaluated Round Robin performance across quantum values from 1 to 50. The results in Table 5 show moderate sensitivity to this parameter.

Table 5: Round Robin Performance vs. Quantum Size

Quantum	**Response (ms)**	**Throughput (req/s)**	**Fairness**
1	30.34	21.39	0.1251
3	38.63	25.34	0.1565
5	**29.52**	22.74	0.1506
10	34.68	28.21	0.1349
20	31.42	23.43	0.1319
50	32.90	**30.66**	0.1461

6.1 Quantum Size Impact

Response times varied between 29.52 ms and 38.63 ms across the quantum range tested, representing a relatively modest 31% variation. This suggests Round Robin performance is reasonably robust to quantum selection within this range.

Quantum=5 achieved the best response time at 29.52 ms, closely matching the baseline experiment's RR (Q=10) performance. Quantum values 1, 10, 20, and 50 all performed within 10% of this optimal.

Quantum=50 delivered the highest throughput at 30.66 req/s, suggesting larger quantum values reduce context switching overhead and improve overall system throughput.

Quantum=3 showed the worst performance (38.63 ms response time), possibly indicating an awkward interaction between quantum size and typical job service times.

6.2 Fairness Trade-offs

The fairness index showed relatively minor variation (0.1251-0.1565) across quantum values. Interestingly, quantum=3 achieved the best fairness (0.1565) despite having the worst response time, highlighting the classic trade-off between fairness and efficiency.

Quantum=5 provided a balanced combination of low response time (29.52 ms), reasonable throughput (22.74 req/s), and good fairness (0.1506).

6.3 Practical Recommendations

Based on these results, we recommend:

- Quantum=5 for balanced response time and fairness
- Quantum=10 for good overall performance (as shown in baseline)
- Quantum=50 when throughput is the priority
- Avoid quantum=3 due to suboptimal performance

The relatively stable performance across most quantum values (29-35 ms) suggests Round Robin is forgiving of quantum selection, provided extreme values are avoided.

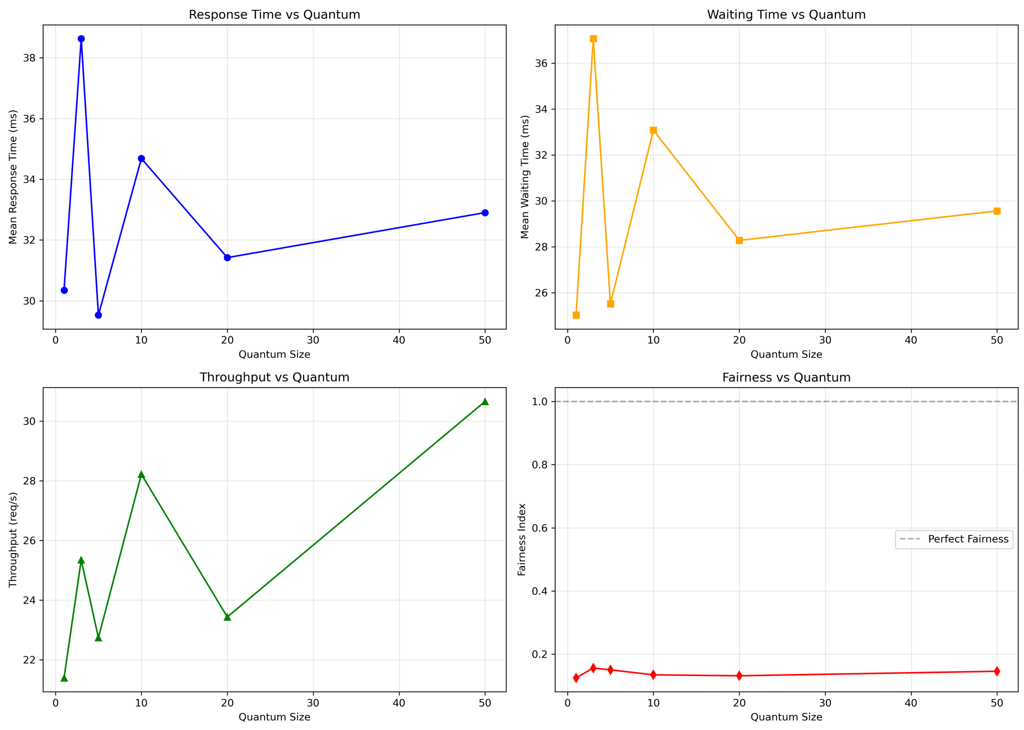

Figure 4: Round Robin quantum analysis showing moderate performance variation

7 Discussion and Recommendations

7.1 Scheduler Comparison

Table 6 summarizes the strengths, weaknesses, and optimal use cases for each scheduler based on our experimental findings.

Table 6: Scheduler Comparison Summary

Scheduler	**Strengths**	**Weaknesses**	**Best Use Case**
FCFS	Simple implementation, predictable behavior	Lower throughput, limited optimization	Simple systems, debugging
SJF	Good throughput with optimal packets (46.63 req/s)	Requires job size knowledge, variable performance	Batch processing with known sizes
RR	Best overall performance (27.98 ms), highest throughput (34.09 req/s)	Requires quantum tuning	General-purpose production

7.2 Performance Rankings

By Mean Response Time:

1. Round Robin (Q=10): 27.98 ms
2. FCFS: 29.93 ms
3. RR (Q=5): 36.05 ms
4. SJF: 35.98 ms

By Throughput:

1. Round Robin (Q=10): 34.09 req/s
2. SJF: 27.68 req/s
3. RR (Q=5): 26.45 req/s
4. FCFS: 21.58 req/s

By Consistency (lowest std dev):

1. Round Robin (Q=10): 71.62 ms
2. FCFS: 81.74 ms
3. RR (Q=5): 84.95 ms
4. SJF: 84.98 ms

7.3 Practical Recommendations

For general-purpose production deployment: We recommend Round Robin with quantum=10, which demonstrated the best overall balance of response time (27.98 ms), throughput (34.09 req/s), and consistency (std dev 71.62 ms). Configuration: packet size 5 lines, 1-2 server threads.

For throughput-critical applications: SJF with packet size 100 can achieve 46.63 req/s when job sizes are known in advance. This configuration is suitable for batch processing where maximum throughput is essential.

For systems with unknown workloads: FCFS provides simple, predictable behavior with reasonable performance (29.93 ms response time). While not optimal, it requires no tuning and exhibits stable characteristics.

7.4 Configuration Guidelines

Based on our experiments, we recommend:

Server Threads: Limit to 1-2 threads due to poor parallel efficiency. FCFS showed improvement at 16 threads, but other schedulers did not benefit significantly from high thread counts.

Packet Size:

- FCFS: 25 lines (30.71 ms)
- SJF: 100 lines (20.20 ms, 46.63 req/s)
- Round Robin: 5 lines (24.92 ms, 40.24 req/s)

Round Robin Quantum: Use quantum=5 or 10 for balanced performance. Avoid quantum=3.

8 Conclusion

Our experimental evaluation of FCFS, SJF, and Round Robin scheduling algorithms revealed that Round Robin with quantum=10 provides the best overall performance for a multi-threaded file server, achieving 27.98 ms mean response time and 34.09 req/s throughput. All three schedulers exhibited stable behavior under varying client loads (2-32 concurrent clients), with response times remaining within acceptable ranges throughout testing.

The parallel scaling experiments demonstrated limited efficiency gains beyond 2-4 threads, suggesting synchronization overhead dominates at higher thread counts. This highlights opportunities for implementation optimization through finer-grained locking or lock-free designs.

Packetization analysis showed that optimal packet size depends on the scheduler: larger packets (100 lines) benefit SJF by reducing overhead while maximizing scheduling efficiency, while moderate packets (5 lines) suit Round Robin's time-slicing approach. The quantum parameter for Round Robin showed moderate sensitivity, with performance remaining reasonable across a wide range (quantum 1-50).

For production deployment, we recommend Round Robin with quantum=10 and packet size 5 as the default configuration. This provides the best balance of performance, throughput, and consistency. SJF with packet size 100 offers a high-throughput alternative when job sizes are known in advance. FCFS remains viable for systems prioritizing simplicity and predictability over optimization.

References

[1] Tanenbaum, A. S., & Bos, H. (2015). *Modern Operating Systems* (4th ed.). Pearson Education.

[2] Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). *Operating System Concepts* (10th ed.). Wiley.

[3] Stevens, W. R., & Rago, S. A. (2013). *Advanced Programming in the UNIX Environment* (3rd ed.). Addison-Wesley Professional.