Dynamic Scheduling Without Real Time Requirement.

MOHAMMAD ASHRAFUZZAMAN SIDDIQI

What is Scheduling?

- Allocation of resources to tasks over time
- Goals: Optimize throughput and latency, and ensure fair resource usage.

Type of scheduling

- Dynamic Scheduling
- Static Scheduling

Static Scheduling

- Pre-determined resource allocation
- Schedule created at compile time

Used in:

- Embedded Systems
- Batch Processing Systems
- Inflexibility to dynamic changes

Dynamic Scheduling

Dynamic scheduling refers to a process in which tasks and resources are assigned in real-time, either after compilation or during the execution of the system.

- Adaptability and real-time decision-making
- Greater flexibility and performance
- Hardware-based approach

Key Algorithms

- Round Robin (RR)
- Shortest Remaining Time First (SRTF)
- Multilevel Feedback Queue (MLFQ)

Dynamic Scheduling Without Real time Requirements

Dynamic scheduling without a real-time requirement can play an important role in enhancing the overall performance of a computing environment.

When deadlines are of no importance Dynamic scheduling helps to:

• **Maximized throughputs:**	**Balancing the workload distribution**
	INCREASES TASK COMPLETION RATE
• **Optimize Resource Utilization:**	**Optimizes use of CPU, memory, and I/O devices.**
	Minimizes idle times.
• **Improve Responsiveness:**	**Prioritizes tasks requiring quick responses.**
	Enhances user experience.

Dynamic Scheduling Scheduling Algorithms

Round Robin (RR): **Preemptive, fixed time quantum for each process**.

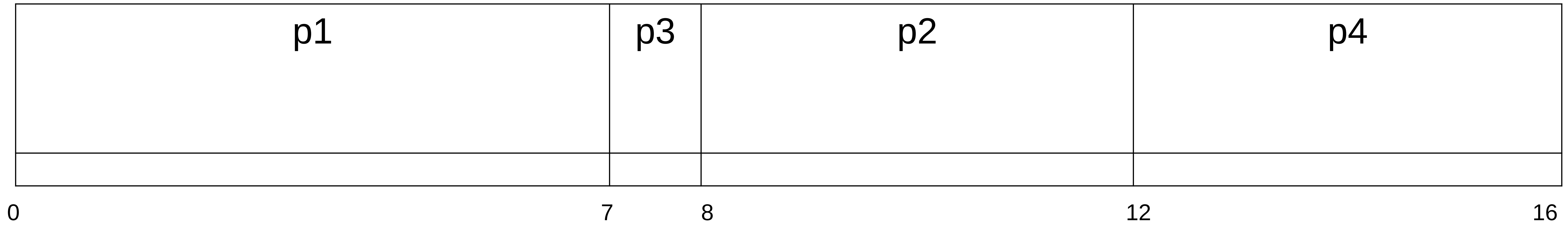
- **Shortest Remaining Time First (SRTF):** SRTF **IS a PREEMTIVE SCHEDULING ALGORITHM**

  SJF **IS NONPREEMTIVE SCHEDULING ALGORITHMS**

- **Multilevel Feedback Queue (MLFQ):** MLFQ is a scheduling algorithm that uses multiple queues with different priority levels. Processes can move between queues based on their behavior and CPU burst characteristics.

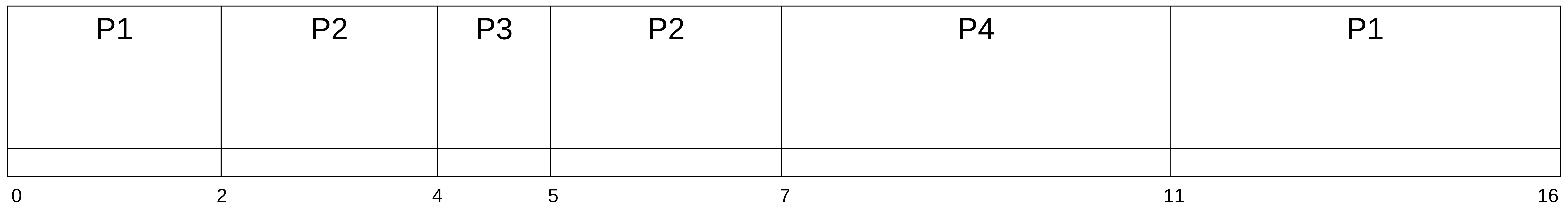
SJN non preemtive

Process	Arrivaltime	CPUtime
P1	0	7
P2	2	4
P3	4	1
P4	5	4



The average waiting time is calculated as:
Average waiting time = (0 + 6 + 3 + 7)/4 = 4

PREEMTIVE SCHEDULING ALGORITHM



Average waiting time = (9 + 1 + 0 + 2)/4
=3

The SRTF algorithm further reduces the average waiting time compared to the non-preemptive SJF algorithm by dynamically selecting the process with the shortest remaining time.

Round Robin (RR)

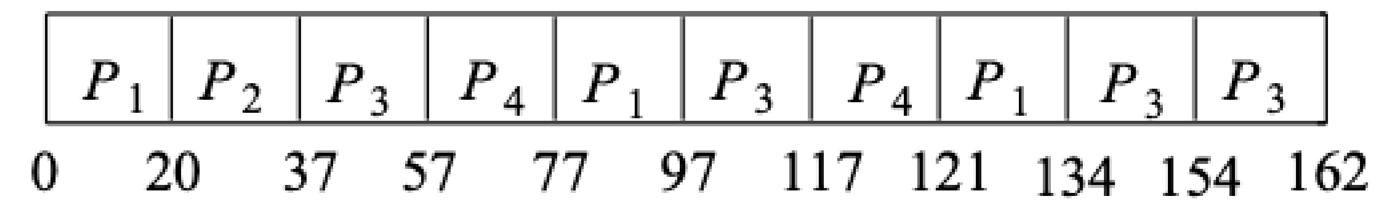
Round Robin scheduling is a preemptive approach in which each process in the system is assigned a fixed amount of quantum time, often between 10-100 milliseconds.

- Cycles through processes, allocating CPU time.
- Fairness and simplicity

Example of RR with time quantum = 20

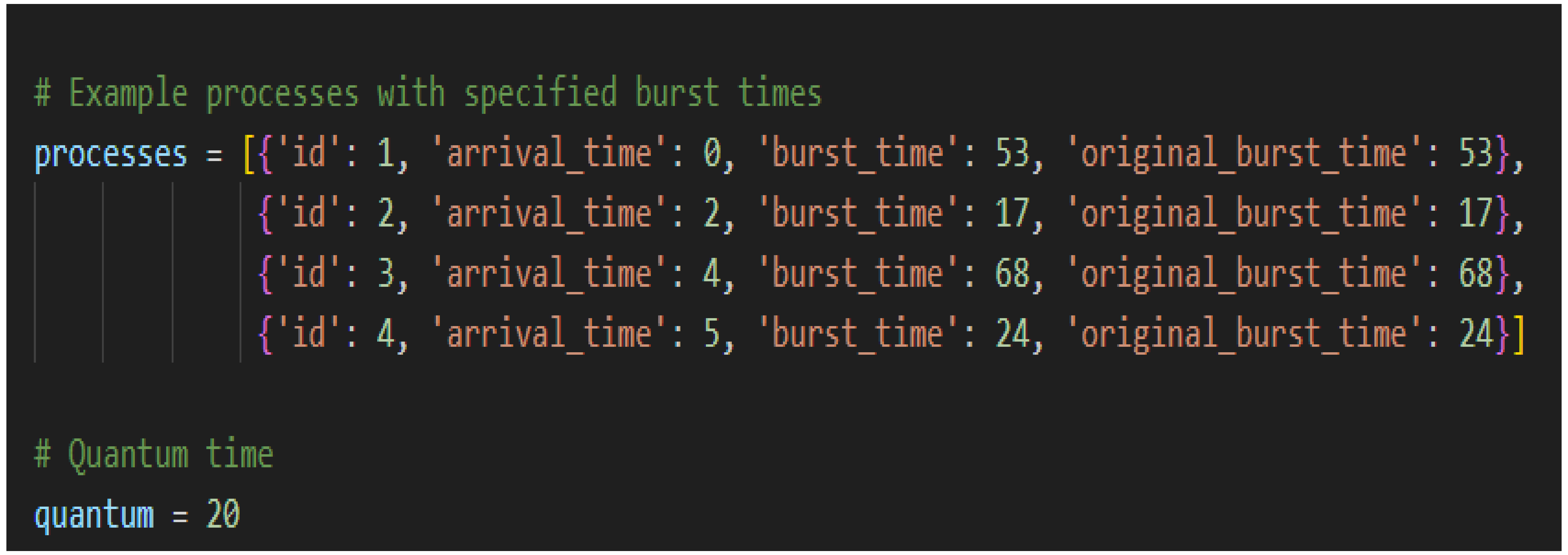
Process	CPU times
P_1	53
P_2	17
P_3	68
P_4	24

- The Gantt chart is:



Round Robin (RR) Python Implementation

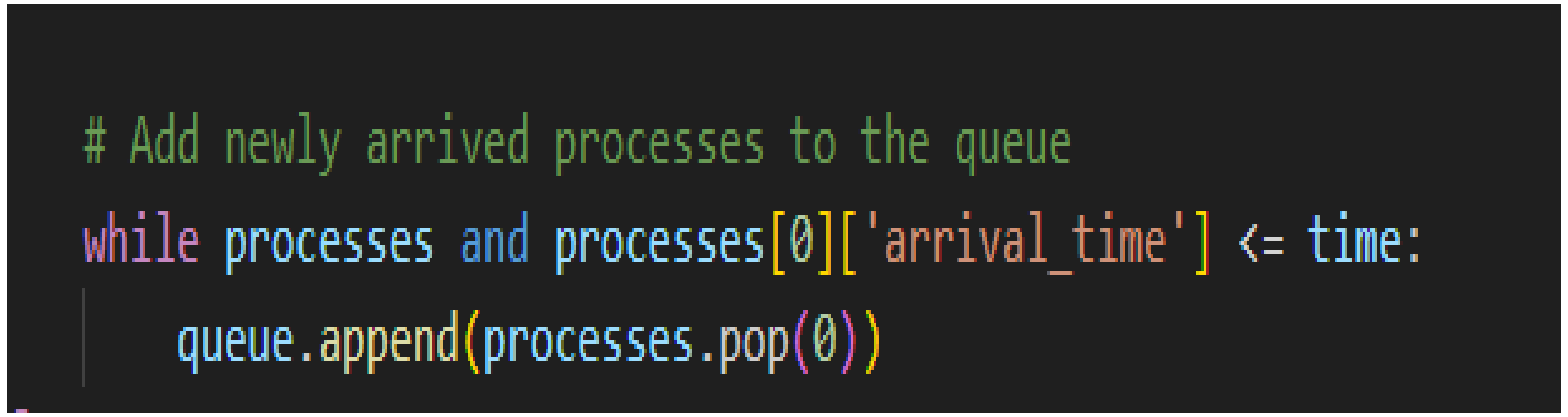
- Initialization of process and queue.
- Process sorted by arrival time,
- New Processes are dynamically added to the queue
- Each process receives the same quantum.



```python
# Example processes with specified burst times
processes = [{'id': 1, 'arrival_time': 0, 'burst_time': 53, 'original_burst_time': 53},
             {'id': 2, 'arrival_time': 2, 'burst_time': 17, 'original_burst_time': 17},
             {'id': 3, 'arrival_time': 4, 'burst_time': 68, 'original_burst_time': 68},
             {'id': 4, 'arrival_time': 5, 'burst_time': 24, 'original_burst_time': 24}]

# Quantum time
quantum = 20
```

Processes and quantum time definition.

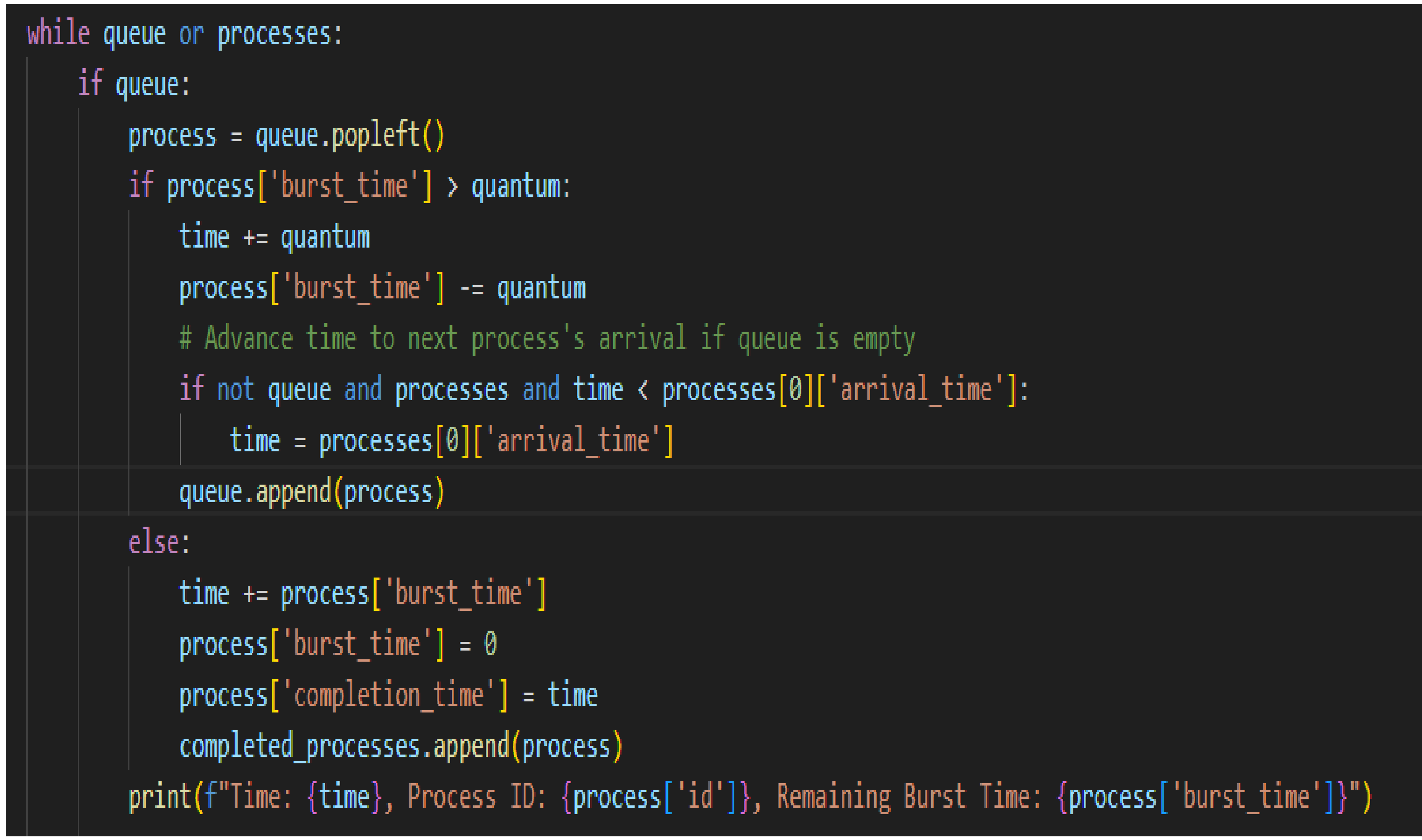


```python
# Add newly arrived processes to the queue
while processes and processes[0]['arrival_time'] <= time:
    queue.append(processes.pop(0))
```

Dynamically Adding to the Queue

Round Robin (RR) Python Implementation

- Completed Processes
- If the burst time of a process is greater than the quantum, it's partially processed, and the remaining burst time is updated.
- Partially completed processes are added to the back of the queue.
- If the burst time of a process is less than or equal to the quantum, the process is completed, and its completion time is recorded.



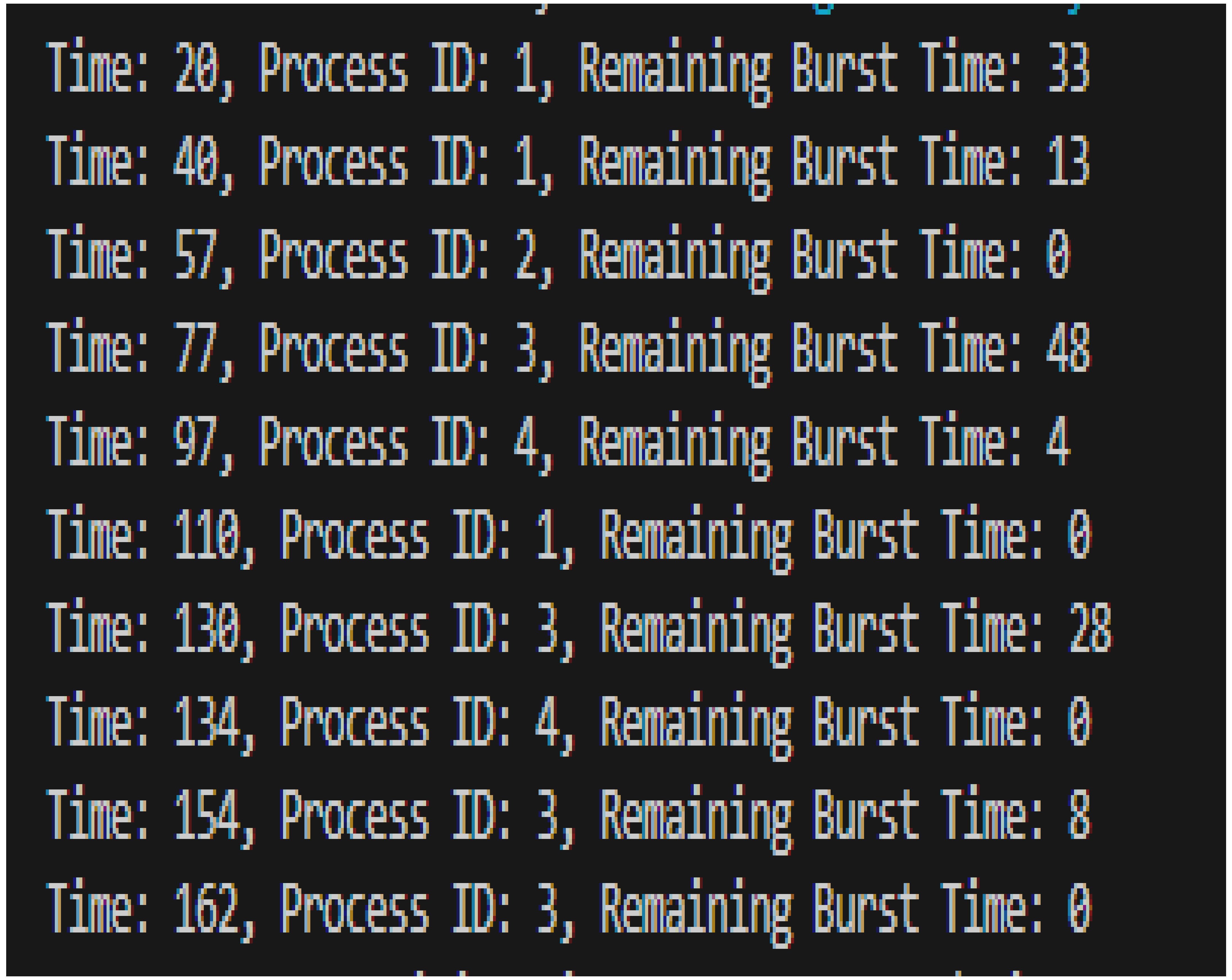
```python
while queue or processes:
    if queue:
        process = queue.popleft()
        if process['burst_time'] > quantum:
            time += quantum
            process['burst_time'] -= quantum
            # Advance time to next process's arrival if queue is empty
            if not queue and processes and time < processes[0]['arrival_time']:
                time = processes[0]['arrival_time']
            queue.append(process)
        else:
            time += process['burst_time']
            process['burst_time'] = 0
            process['completion_time'] = time
            completed_processes.append(process)
        print(f"Time: {time}, Process ID: {process['id']}, Remaining Burst Time: {process['burst_time']}")
```

Processing of Tasks.

Round Robin (RR)-Python Results

- **The results Shows Round robin ensures that each process receives a fair share of CPU time without being starved.**

- **The consistent intervals in the results validate that the Round Robin scheduling evenly distributes CPU resources, which helps maintain system responsiveness and performance.**

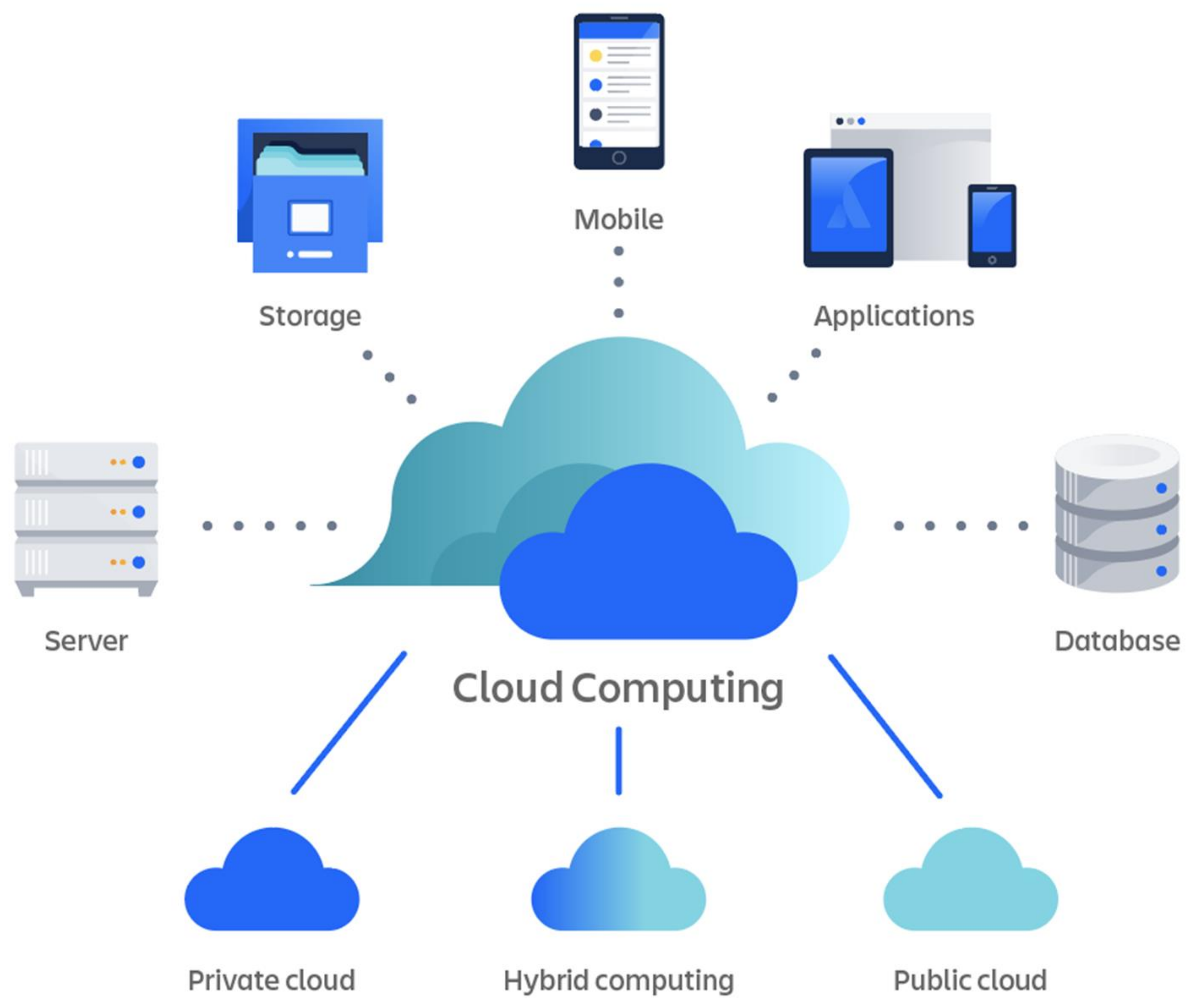


```
Time: 20, Process ID: 1, Remaining Burst Time: 33
Time: 40, Process ID: 1, Remaining Burst Time: 13
Time: 57, Process ID: 2, Remaining Burst Time: 0
Time: 77, Process ID: 3, Remaining Burst Time: 48
Time: 97, Process ID: 4, Remaining Burst Time: 4
Time: 110, Process ID: 1, Remaining Burst Time: 0
Time: 130, Process ID: 3, Remaining Burst Time: 28
Time: 134, Process ID: 4, Remaining Burst Time: 0
Time: 154, Process ID: 3, Remaining Burst Time: 8
Time: 162, Process ID: 3, Remaining Burst Time: 0
```

Applications

Cloud Computing

- Effective resource management and load balancing.
- Dynamic task allocation to virtual machines.
- Improved system performance and cost reduction.

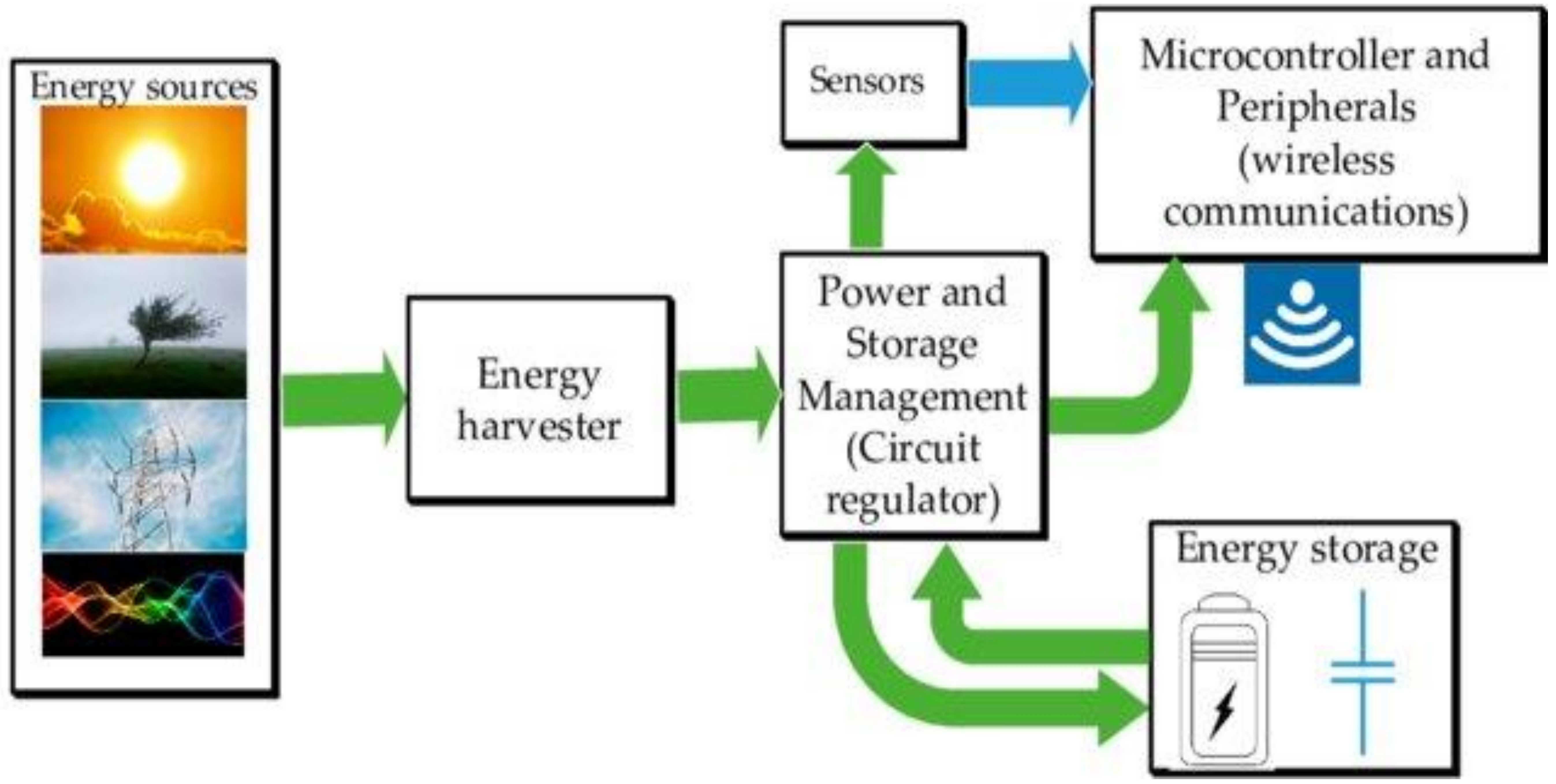


Manufacturing Systems

- Algorithms adjust task schedules based on current shop floor conditions
- Enhances manufacturing efficiency by minimizing downtime.

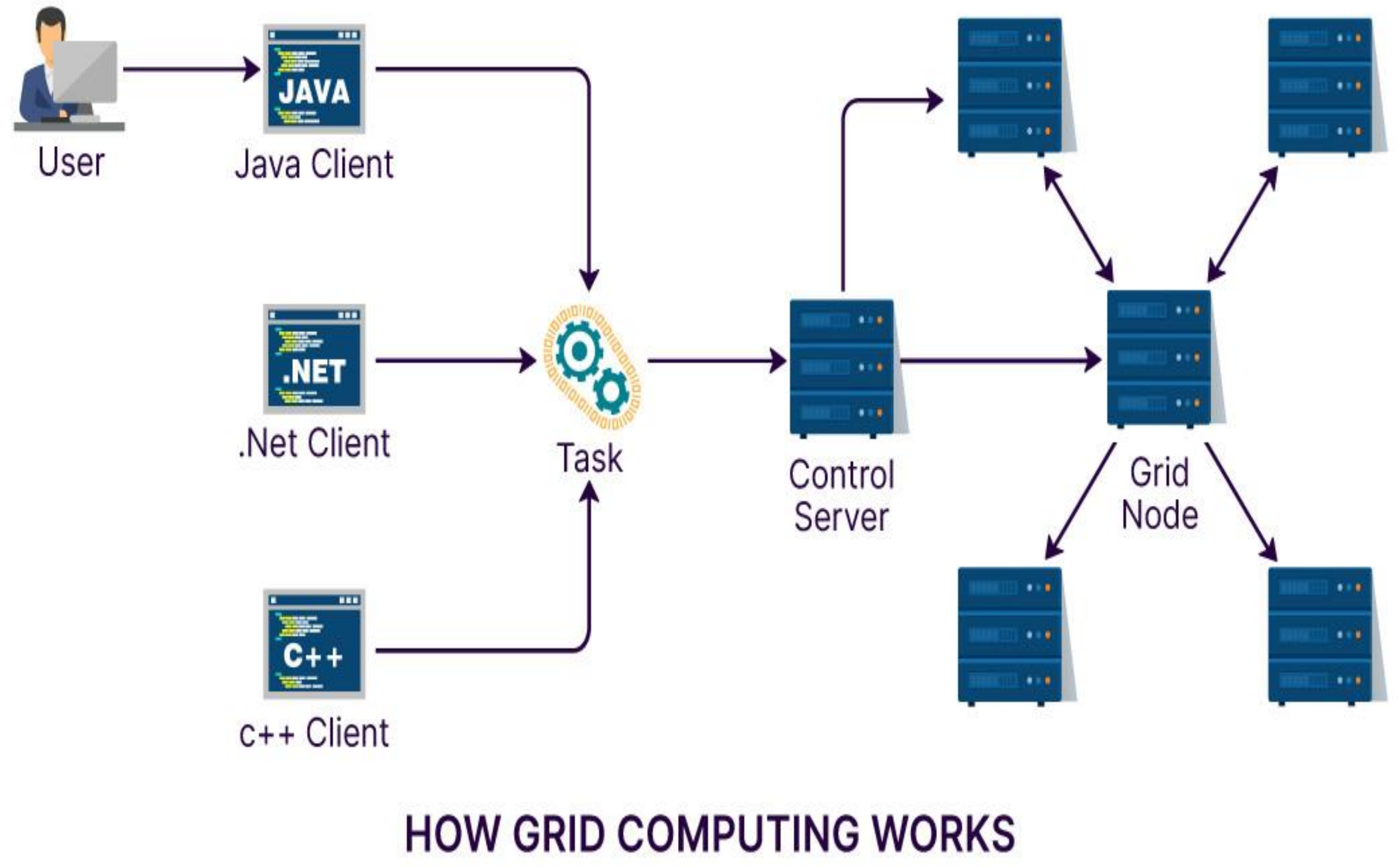
Energy Harvesting Systems

- Activates energy harvesting modules as needed.
- Balances energy harvesting and consumption for efficiency.


Energy sources
Energy harvester
Power and Storage Management (Circuit regulator)
Sensors
Microcontroller and Peripherals (wireless communications)
Energy storage

Grid Computing

- Prioritizes tasks based on their criticality and resource requirements
- Ensures efficient execution and resource utilization.
- Optimizes complex scientific computations.



Conclusion

- This paper examines dynamic scheduling algorithms, specifically Round Robin (RR) and Shortest Remaining Time First (SRTF), in non-real-time systems. These algorithms significantly improve resource management and efficiency across various applications such as cloud computing, manufacturing, energy harvesting, IoT, and grid computing. By dynamically adapting to workloads and conditions, RR and SRTF ensure optimal performance and cost-effectiveness, demonstrating their critical importance in modern technological environments.