Future Enhancements (Suggestions for future improvements or additional features that could be implemented)

While the implemented module for simulating Multilevel Queue (MLQ) CPU Scheduling effectively mitigates the issue of CPU starvation, there are several potential areas for future enhancements and additional features. Here are some suggestions for further improving the module.

Dynamic Priority Adjustment

Currently, the module promotes lower-priority processes to higher-priority queues at fixed intervals. However, in a real-world scenario, the priority of a process may change based on its behavior or external factors. Implementing a mechanism for dynamically adjusting the priority of processes based on their resource requirements, execution time, or other relevant factors would enhance the flexibility and adaptability of the module.

Aging Mechanism

To further prevent CPU starvation, an aging mechanism can be introduced. This mechanism gradually increases the priority of processes that have been waiting in lower-priority queues for an extended period. By gradually boosting the priority of long-waiting processes, the module can ensure that they eventually get access to the CPU, even if higher-priority queues are continuously occupied.

Preemption Policies

Currently, the module does not support preemption, which means once a process is assigned to a queue, it continues executing until it completes or voluntarily yields the CPU. Adding support for preemption policies would allow higher-priority processes to preempt lower-priority processes and gain immediate access to the CPU. This would further improve responsiveness and fairness in resource allocation.

Dynamic Queue Allocation

Instead of fixed priority levels, introducing a mechanism for dynamically allocating queues based on system load or process characteristics would enhance the adaptability of the module. For example, during periods of high system load, additional queues with higher priorities could be dynamically created to handle critical processes. Conversely, during periods of low load, unnecessary queues could be temporarily deactivated to optimize resource utilization.

Advanced Scheduling Algorithms

While the implemented module uses a basic MLQ CPU Scheduling algorithm, there are more advanced scheduling algorithms available that could be considered for future enhancements. Algorithms such as Round-Robin with Time Quantum and Feedback Scheduling provide additional capabilities for managing CPU resources and optimizing performance. Implementing these algorithms as options within the module would allow users to choose the most suitable scheduling strategy based on their specific requirements.

Visualization and Monitoring

Adding visualization capabilities to the module would provide a graphical representation of the scheduling process, making it easier to understand and analyze resource allocation. Additionally, incorporating monitoring features to track and display system metrics such as CPU utilization, waiting times, and queue lengths would enable users to gain insights into system performance and identify potential bottlenecks or areas for improvement.

Support for Real-Time Processes

Real-time processes have strict timing requirements and need to be executed within specific deadlines. Enhancing the module to support real-time processes with priority-based scheduling policies would enable it to handle time-critical applications more effectively.

Integration with Other System Components

To make the module more versatile and adaptable, integration with other system components such as I/O scheduling or memory management can be considered. This integration would enable a more comprehensive and coordinated approach to resource allocation within the system.

By implementing these future enhancements and additional features, the MLQ CPU Scheduling module can further improve its performance, adaptability, and overall effectiveness in managing CPU resources and mitigating the issue of CPU starvation. These enhancements would cater to a wider range of scenarios and provide users with more control over resource allocation strategies.