❖ Proposed new MPTCP scheduler:

Multipath TCP (MPTCP) is an extension of TCP that allows a TCP connection to use multiple paths to improve performance, reliability, and resource utilization. The scheduler in MPTCP determines how data is distributed across the available paths. Proposing a new MPTCP scheduler involves designing a mechanism that optimally utilizes the multiple paths in a way that benefits the overall performance of the connection.

Here are some considerations and ideas for a proposed MPTCP scheduler:

1. Path Monitoring:
- Implement a robust path monitoring mechanism to continuously assess the quality and availability of each path.
- Consider factors such as latency, throughput, packet loss rate, and available bandwidth to dynamically adjust the weights assigned to each path.

2. Dynamic Path Weighting:
- Develop a dynamic weighting algorithm that adjusts the distribution of traffic among paths based on real-time path conditions.
- For example, paths with lower latency or higher throughput might be assigned higher weights.

3. Load Balancing:
- Implement load balancing strategies to distribute the traffic load more evenly across paths.
- Load balancing can help prevent congestion on a particular path and utilize the available bandwidth more efficiently.

4. **Resilience and Fault Tolerance:**
- Design the scheduler to be resilient to path failures and able to quickly reroute traffic to available paths.
- Implement mechanisms to detect and react to path failures, minimizing the impact on the overall connection.

5. User Configurability:
- Allow users or network administrators to configure scheduler parameters based on their specific requirements and network conditions.
- Provide flexibility in terms of specifying path priorities, weights, and other relevant parameters.

6. Congestion Control:
- Coordinate with the existing congestion control mechanisms to ensure that the scheduler does not exacerbate congestion issues.
- Consider congestion-aware algorithms to adaptively adjust the traffic distribution in response to varying congestion levels.

7. Cross-Layer Optimization:
- Explore opportunities for collaboration with lower layers of the network stack, such as link-layer protocols, to enhance overall performance.

- Coordinate with network protocols to obtain more accurate information about the state of the paths.

8. Simulation and Testing:
- Simulate the proposed scheduler in various network scenarios to assess its performance under different conditions.
- Conduct thorough testing to validate the scheduler's effectiveness, stability, and compatibility with existing MPTCP implementations.

❖ Comparing a new Multipath TCP (MPTCP) scheduler with existing:

Comparing a new Multipath TCP (MPTCP) scheduler with existing schedulers involves evaluating their performance, efficiency, and adaptability in various network conditions. Below are key aspects to consider when comparing a new MPTCP scheduler with existing ones:

1. Path Utilization:
- **Existing Scheduler:** Evaluate how well the existing scheduler utilizes available paths in terms of balancing traffic and efficiently utilizing available bandwidth.
- **New Scheduler:**Demonstrate improvements in path utilization, possibly through dynamic path weighting, load balancing, or other optimization strategies.

2. Adaptability to Network Conditions:
- **Existing Scheduler:** Assess how well the existing scheduler adapts to changing network conditions, such as varying latency, throughput, and packet loss.
- **New Scheduler:** Highlight adaptive mechanisms that enable the new scheduler to dynamically adjust its behavior based on real-time path performance.

3. Fault Tolerance:
- **Existing Scheduler**: Examine how the existing scheduler handles path failures and reroutes traffic in the event of path unavailability.
- **New Scheduler:** Showcase improvements in fault tolerance, rapid path detection, and seamless rerouting to maintain connection reliability.

4. Congestion Control:
- **Existing Scheduler:** Analyze the congestion control mechanisms integrated with the existing scheduler and their impact on overall performance.
- **New Scheduler:** Demonstrate how the new scheduler collaborates with congestion control to prevent congestion, adapt to varying conditions, and optimize overall throughput.

5. User Configurability:
- **Existing Scheduler:** Evaluate the level of configurability provided to users or administrators in terms of specifying path priorities, weights, and other parameters.
- **New Scheduler:** Highlight any enhancements in user configurability, allowing fine-tuning of scheduler parameters to match specific network requirements.

6. Simulation Results:

- **Existing Scheduler:** Consider existing simulation results and performance evaluations conducted on the current MPTCP scheduler.
- **New Scheduler:** Present simulation results and comparative analyses, demonstrating the superiority or unique advantages of the proposed scheduler.

7. Cross-Layer Optimization:

- **Existing Scheduler:** Investigate any collaboration with lower layers of the network stack and its impact on overall performance.
- **New Scheduler:** Showcase any cross-layer optimization strategies that enhance coordination with lower-layer protocols for improved efficiency.

8. Community Adoption and Feedback:

- **Existing Scheduler**: Consider the existing scheduler's adoption within the MPTCP community and any feedback from users or developers.
- **New Scheduler**: Engage with the MPTCP community for feedback, collaboration, and potential integration into standard MPTCP implementations.

9. Ease of Integration:

- **Existing Scheduler:** Assess how easily the existing scheduler can be integrated into different MPTCP implementations.
- **New Scheduler:** Emphasize the ease of integration and compatibility with existing MPTCP implementations or frameworks.

10. Scalability:

- E**xisting Scheduler:** Examine how well the existing scheduler scales with the number of available paths and the complexity of network topologies.
- **New Scheduler:** Highlight any scalability improvements, ensuring efficient operation in scenarios with a large number of paths.

❖ Optimization of new proposed MPTCP scheduler:

Optimizing a proposed Multipath TCP (MPTCP) scheduler involves refining its design and implementation to enhance performance, efficiency, and adaptability. Here are some optimization strategies you may consider for a new MPTCP scheduler:

1. Path Selection Algorithms:

- Optimize the algorithm used for path selection based on real-time conditions. Consider factors like latency, throughput, and packet loss in determining the most suitable path for data transmission.

2. Dynamic Path Weighting:

- Implement dynamic path weighting mechanisms that adapt to changing network conditions. Use feedback from ongoing connections to adjust weights in real-time, optimizing traffic distribution.

3. Load Balancing Strategies:

- Refine load balancing techniques to distribute traffic more evenly across available paths. Explore adaptive load balancing approaches that react to changes in path conditions or network topology.

4. Congestion-Awareness:

- Enhance congestion-awareness by implementing algorithms that respond to congestion signals promptly. Consider collaboration with existing congestion control mechanisms to ensure optimal performance.

5. Predictive Analytics:

- Explore the use of predictive analytics to anticipate changes in network conditions. Predictive models can help the scheduler proactively adjust its strategy before performance degradation occurs.

6. Machine Learning Integration:

- Investigate the integration of machine learning algorithms for dynamic decision-making. Machine learning models can learn from historical data and adapt the scheduler's behavior to optimize performance.

7. Cross-Layer Optimization:

- Collaborate with lower layers of the network stack for better coordination. Optimize interactions with link-layer protocols to leverage more accurate and timely information about the state of individual paths.

8. Fast Path Failure Detection:

- Optimize the detection and response to path failures. Implement mechanisms for rapid detection of path unavailability and seamless rerouting of traffic to healthy paths.

9. Parallelization:

- Explore opportunities for parallelization to leverage the capabilities of modern multi-core processors. Optimize the scheduler's processing pipeline to handle multiple paths concurrently.

10. User Configurability and Feedback:

- Provide users or administrators with more granular control over scheduler parameters. Implement mechanisms for user feedback to adapt the scheduler's behavior based on specific use cases and preferences.

11. Asynchronous Operation:

- Design the scheduler to operate asynchronously when possible. Asynchronous processing can improve responsiveness and resource utilization in scenarios where paths have varying latencies or delays.

12. Scalability Considerations:

- Optimize the scheduler for scalability, ensuring efficient operation as the number of paths or connections increases. Consider optimizations for both large-scale deployments and scenarios with a limited number of paths.

13. Efficient Resource Utilization:

- Optimize resource consumption, such as memory and CPU usage. Ensure the scheduler is lightweight and efficient to avoid unnecessary overhead.

14. Simulation and Testing:

- Continuously simulate and test the scheduler in various network scenarios to identify areas for improvement. Use realistic test cases to validate the optimization strategies and measure their impact on performance.

15. Documentation and Reporting:

- Clearly document the optimization strategies employed, their impact on performance, and any trade-offs made during the optimization process. Provide comprehensive reports on the scheduler's behavior in different scenarios.