Kritische Analyse der OSMP-Implementierung

1. Kritische Abschnitte im System

FreeSlotQueue

- Zugriff auf head und tail ist kritisch
- Muss synchronisiert werden, um doppelte Slotvergabe zu verhindern
- Geschützt durch:
    - sem_t mutex für exklusiven Zugriff
    - sem_t sem_slots zum Steuern der Gesamtanzahl verfügbarer Slots

Mailboxen

- Pro Prozess eine Mailbox (FIFO-Warteschlange)
- Kritische Abschnitte: head, tail, Verkettung via next
- Synchronisation durch:
    - sem_space (Zähler für freie Plätze)
    - sem_data (Zähler für Nachrichten zur Abholung)
    - mutex (Binärsemaphor für Zugriffsschutz)

Nachrichtenslots (MessageSlot[])

- Werden beim Send beschrieben, beim Recv gelesen
- Keine parallelen Schreib-/Lesezugriffe durch Sperrlogik möglich
- Zugriff immer eindeutig: Slot ist entweder "frei", "belegt" oder "wird zurückgegeben"

---

2. Synchronisationsmechanismen & Entscheidungen

Semaphore

- Zählende Semaphoren: Ressourcenverwaltung (Slots, Platz, Daten)
- Binäre Semaphore: Zugriffsschutz wie bei Mutexen (z. B. Mailbox-Zugriff)
- Vorteil: POSIX-Semaphore funktionieren zwischen Prozessen via Shared Memory

Kein Einsatz von pthread_mutex_*

- Threadsynchronisation nicht notwendig, da OSMP prozessbasiert
- POSIX-Semaphore reichen zur Interprozess-Synchronisation aus

FIFO in Mailbox

- Gewährleistet, dass Nachrichten in **Ankunftsreihenfolge** verarbeitet werden
- Ermöglicht Fairness zwischen Prozessen

---

3. Risikoanalyse

Race Conditions

- **Verhindert durch:**
    - Semaphore für jeden kritischen Abschnitt
    - Keine ungeschützten Schreib-/Lesezugriffe auf Queue- und Mailboxindizes

Deadlocks

- **Potenzielle Ursache:** zyklisches Warten z. B. bei sem_wait(sem_space) + sem_wait(sem_slots)
- **Vermeidung:**
    - Klare Sperrreihenfolge: immer erst sem_wait(sem_slots) → dann mutex
    - Keine zyklischen Abhängigkeiten im Code

Starvation

- Theoretisch möglich, praktisch unwahrscheinlich bei Fairness in Semaphoren
- FIFO-Verkettung in Mailbox sorgt für gerechte Verarbeitung

Slot-Verlust

- Verhindert durch:
    - Pflicht zur Slot-Rückgabe nach erfolgreichem Recv
    - FreeSlotQueue mit Ringpuffer garantiert, dass kein Slot verloren gehen kann

---

4. Bewertung der Robustheit

Aspekt	**Status**
Synchronisation	Vollständig
Fairness	Gewährleistet

Aspekt	Status
Ressourcen-Management	Effizient, fehlerfrei
Deadlock-Vermeidung	Strukturell umgesetzt
Erweiterbarkeit	Hoch

Fazit

Die OSMP-Implementierung ist synchronisiert, blockierend, aber sicher. Kritische Abschnitte sind über Semaphore geschützt. Die Struktur verhindert Deadlocks und Race Conditions durch durchdachte Zugriffsreihenfolgen. Die Modularität erlaubt eine einfache Erweiterung um Funktionen wie Barrier oder nicht-blockierendes Senden.