Betriebssysteme, Übungsblatt 2

Die Aufgaben wurden an einem Windows-Rechner in VS Code mit der Programmiersprache C bearbeitet. Für jede Aufgabe wurden 1000 Nachrichten von einem Sender an einen Receiver verschickt und die Latenz aufgezeichnet. Der 95% Confidenzintervall wurde anhand einer studentschen t-Verteilung errechnet.

1.

In der ersten Aufgabe habe ich die Latenzen zwischen zwei Threads innerhalb eines Prozesses gemessen. Wie zu erwarten, waren die Latenzen für dieses einfache Beispiel sehr niedrig.

```
Sender: Minimal Latency = 100 ns
Sender: 95% Confidence Interval for Minimal Latency = 600 ns
PS C:\Users\Jason\projects\OperatingSystems>
```

2.

Auch hier wurden Latenzen innerhalb eines Prozesses gemessen. Im Vergleich zur ersten Aufgabe hat die Verwendung von Semaphores zu leicht höheren Latenzen geführt. Dieses Ergebnis liegt vermutlich an dem größeren Overhead von Semaphores.

```
Sender: Minimal Latency = 300 ns
Sender: 95% Confidence Interval for Minimal Latency = 1500 ns
PS C:\Users\Jason\projects\OperatingSystems>
```

3.

In der dritten Aufgabe wurden Latenzen unter der Verwendung von ZeroMQ verglichen zwischen Threads innerhalb und außerhalb eines Prozesses. Die Ergebnisse waren hier sehr unterschiedlich. Die Messungen für Latenzen innerhalb eines Prozesses waren sehr schnell, so dass die kleinste gemessene Latency auf 0 gerundet wurde (Limitierung von CLOCK_MONOTONIC). Auch der 95% Confidenzintervall fiel sehr niedrig aus

Im Vergleich war die Kommunikation zwischen Prozessen deutlich langsamer. Der minimalste Wert war auch hier ziemlich niedrig, aber der 95% Confidenzintervall war deutlich höher. Möglicherweise liegt dies an deutlich mehr Ausreißern im Vergleich zu Latenzen innerhalb eines Prozesses.

```
Minimal Latency: 0 ns
95% Confidence Interval: 500 ns
```

```
Sender: Minimal Latency = 500 ns
Sender: 95% Confidence Interval for Minimal Latency = 12400 ns
```

4.

Für die letzte Aufgabe wurden Latenzen der Kommunikation zwischen Docker-Containern gemessen. Überaschenderweise waren die Ergebnisse hier besser als bei der vorherigen Aufgabe was ich für etwas verwunderlich empfand, da die Verwendung von Dockern meines Wissens nach maximal genauso gut wie Kommunikation ohne Container sein sollte und nicht besser. Vermutlich sind die Unterschiede hier maßgeblich durch Unterschiede in meinen Implementierungen und eventuell durch mehr Outlier in der vorherigen Aufgabe zu erklären

```
Sender: RTT test message received, RTT = 34402 ns
Sender: Minimal Latency = 1368 ns, 95% Confidence Interval = 2966.07 ns
```