Institut für Informationstechnologie (ITEC) Austrian Institute of Techonology (AIT)	SS 2020
Aufgabenblatt 2 – Teil 1 Master – Slave(s) Pattern	

Geben Sie bitte die vollständige Implementierung Im Moodle-Kurs ab. Sie können in Gruppen von bis zu maximal vier Personen arbeiten. Laden Sie auch alle dazugehörigen Dokumente (Sequenzdiagramme, Dokumentation) hoch.

1. Master – Slave Netzwerkkommunikation (Java)

Ziel dieser Aufgabe ist es das Master – (n-)Slave(s) Pattern für das Verteilen von Aufgaben zu implementieren. Hierzu wiederholen Sie die Grundlagen der Java Socket – Programmierung aus dem Kurs WS 2018 Rechnernetze / oder alternativ unter: https://www.javatpoint.com/socket-programming . Verwenden sie als Transportprotokoll TCP. Um das Master – Slave Pattern zu implementieren entwerfen Sie zuerst das Kommunikationsprotokoll. Hierzu sollen zwei Nachrichtentypen, **Initialize**, **Exercise** und **Result**, definiert werden die folgende Protokollfelder beinhalten:

1. Nachrichtentyp: Initialize, Exercise oder Result
2. ID (Nummer) des Slaves, nur bei Initialize
3. Datenlänge (Bytelänge der Daten)
4. Daten (als Byte-Array)

Die Feldlängen sollen korrekt bestimmt und der Protokollverlauf soll als **Sequenzdiagramm** dargestellt werden. Das Masterprogramm soll immer ein festes Port haben, zum Beispiel 9120.

Der Programmablauf soll wie folgt aussehen (angenommen 1 Master und 2 Slaves):

1. (Master) Der Master muss zu erst gestartet werden und auf etwaige Slaves warten (jeder Slave sendet eine **Initialize** Nachricht, Bedingung für das Beenden des Wartemodus ist entweder, dass sich M Slaves gemeldet haben oder aber eine gewisse Zeitspanne verstrichen ist).
2. (Master) Sobald der Wartemodus beendet ist und sich mindestens ein Slave gemeldet hat soll die Verteilung der Aufgabe gestartet werden (hier verwenden Sie einfach Dummy-Daten und füllen die Nachricht mit zufälligen Daten um die Implementierung des Protokolls zu testen).
3. (Slaves) Sobald ein Slave gestartet wird kontaktiert er den Master (mittels der fixierten IP Adresse und Port) und sendet eine **Initialize** Nachricht mit seiner eindeutig zugewiesenen Identifikationsnummer und wartet danach auf weitere Nachrichten vom Master.
4. (Slaves) Erhält ein Slave eine **Exercise** Nachricht, so beginnt er sofort mit der Abarbeitung der darin enthaltenen „Aufgabe“ (zu Testzwecken warten Sie einfach eine definierte Zeitspanne, z. B. eine Sekunde).
5. (Slaves) Ist ein Slave mit seiner Aufgabe fertig wird eine **Result** Nachricht gesendet, die das errechnete Resultat beinhalten. Senden Sie hierzu Testdaten zurück zum Master.

6. (Master) Der Master wartet bis er von allen registrierten Slaves eine Result Nachricht empfangen hat und führt die Ergebnisse zusammen. Für Testzwecke geben Sie die erhaltenen Daten auf der Konsole aus.

Der Server/Slave soll über Kommandozeilenparameter konfiguriert werden (TCP - Port, maximale Anzahl an Slaves, Timeout für den Wartemodus).

Hinweise:

- Überlegen Sie zuerst welche Bytelängen die Felder der Nachrichten haben müssen und wie deren Werte aussehen.
- Zum Veranschaulichen des Protokolls sollen Sequenzdiagramme für jeden Fall erstellt werden.

2. Master - Slave Fault Tolerance (Master)

Erweitern Sie die Funktionalität des Protokolls aus **1.** um den Fall, dass ein Slave seine erhaltene Aufgabe nicht bewältigen kann (aufgrund von Ausfall, fehlenden Ressourcen, etc.). Hierzu erweitern Sie das Kommunikationsprotokoll so, dass der Master nur eine gewisse Zeitspanne auf das Abarbeiten der verteilen Aufgaben wartet (definierbar über die Kommandozeile). Nach Ablauf dieser Zeitspanne sollen nicht erhaltene Resultate auf Slaves verteilt werden, die bereits Ihre Aufgabe erfolgreich erledigt haben. Dieser Vorgang soll so lange wiederholt werden bis zu jeder Teilaufgabe ein Ergebnis vorliegt.

3. Master – Slave Matrixmultiplikation

Erweitern und Testen Sie Ihre Implementierungen aus **1. und 2.** mittels einer Matrixmultiplikation. Hierzu sollen zwei Matrizen $\mathbb{Z}^4 \times \mathbb{Z}^{100}$multipliziert werden. Machen Sie sich klar wie die Matrixmultiplikation parallel berechnet werden kann. Nutzen Sie ihre Implementierungen aus 1. und 2. um das Ergebnis der Matrixmultiplikation verteilt (und parallel) zu berechnen. Zu Beginn soll der Master die Matrizen zufällig mit Integerwerten befüllen. Der Master soll das Ergebnis am Ende ausgeben.