Projektplan: Peer-to-Peer Chat-Programm (Python)

Projektbeschreibung

Im Rahmen des Moduls "Betriebssysteme und Rechnernetze" entwickeln wir ein dezentrales Peer-to-Peer (P2P) Chat-Programm in Python. Das Programm soll es ermöglichen, Textnachrichten und Bilder zwischen Clients zu verschicken und zu empfangen, ohne dass ein zentraler Server verwendet wird. Grundlage ist ein vorgegebenes, proprietäres Chat-Protokoll. Die Kommunikation erfolgt über Sockets, und die Prozesse werden über IPC-Mechanismen verbunden.

Das zugrundeliegende Chat-Protokoll ist **SLCP (Simple Local Chat Protocol)**, das textbasierte Nachrichten, Discovery-Mechanismen via UDP-Broadcast sowie direkte Kommunikation via TCP/UDP unterstützt.

Grobarchitektur

Das System besteht aus drei Hauptprozessen:

- **Benutzer-Schnittstelle (UI-Prozess):** Realisierung einer Kommandozeilenschnittstelle (CLI) zur Eingabe von Befehlen und Anzeige empfangener Nachrichten. Optional Erweiterung um eine GUI.
- **Netzwerk-Kommunikation (Netzwerk-Prozess):** Verantwortlich für den Versand und Empfang von Nachrichten und Bildern. Nutzt TCP/UDP-Sockets basierend auf dem Protokoll.
- **Discovery-Dienst (Discovery-Prozess):** Erkennt und verwaltet verfügbare Peers im Netzwerk. Dieser Dienst läuft nur einmal, auch wenn mehrere Clients gestartet werden.

Die Prozesse kommunizieren über IPC, z.B. Pipes oder Sockets.

Verwendete Technologien

- **Programmiersprache:** Python (Version 3.10+)
- **Entwicklungsumgebung:** VS-Code, PyCharm, optional CLI-Tools
- **Versionskontrolle:** Git (GitHub/GitLab)
- **Bibliotheken:**
    - socket (Netzwerk)
    - multiprocessing (IPC, Prozesssteuerung)
    - threading (ggf. für parallele Aufgaben)
    - configparser (Konfigurationsdatei)
    - base64 (Kodierung von Bildern)
    - Optional: tkinter oder PyQt für GUI

Die Konfigurationsdatei wird im **TOML-Format** gespeichert und enthält folgende Parameter:

- handle – Benutzername des Clients
- port – Lokaler UDP-Portbereich
- whoisport – Port für Discovery-Broadcasts
- autoreply – Automatische Antwortnachricht bei Abwesenheit
- imagepath – Speicherort für empfangene Bilder

Projekt- und Zeitplan

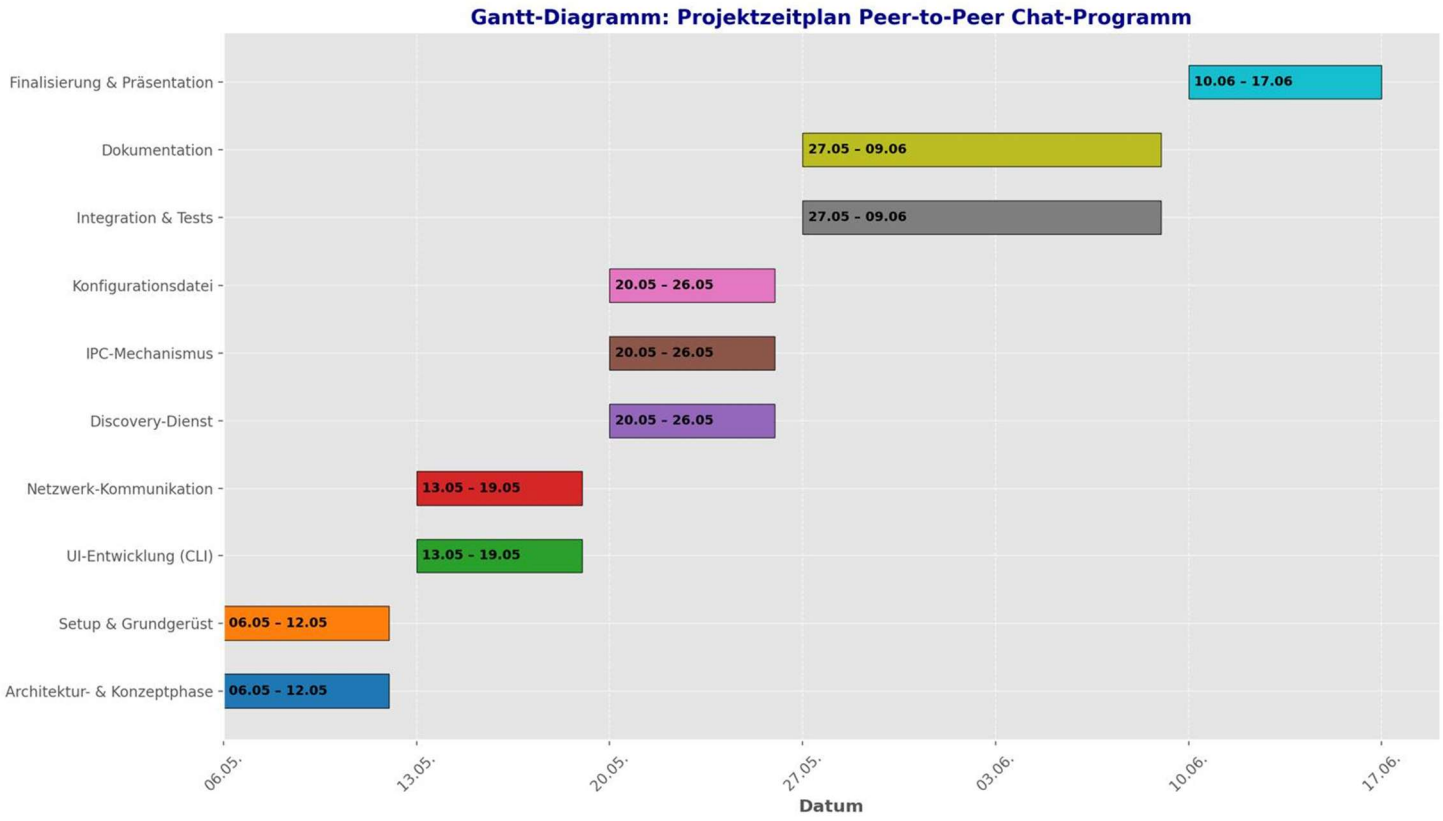
Arbeitspakete und Verantwortlichkeiten

Arbeitspaket	Beschreibung	Hauptverantwortlich	Team-Mitglieder Beteiligung
Architektur- & Konzeptphase	Protokoll klären, Aufgaben planen	Teamleiter: Mohammad Milad	Alle
Setup & Grundgerüst	Git-Repo, Basiscode, Projektstruktur	Noel-Leon	Alle

UI-Entwicklung (CLI)	Entwicklung der Kommandozeilen-Oberfläche	Mulugeta	Mohammad Milad, Nurettin (Testing)
Netzwerk-Kommunikation	Umsetzung von Sockets, Versand & Empfang	Nurettin	Noel-Leon (Review & Testing)
Discovery-Dienst	Implementierung des Peer-Discovery-Mechanismus	Artian	Mulugeta (Testing & Review)
IPC-Mechanismus	Kommunikation zwischen den Prozessen	Mohammad Milad	Nurettin
Konfigurationsdatei	Erstellung & Bearbeitung der Konfigurationsdatei	Noel-Leon	Artian
Integration & Funktionstests	Zusammenführung aller Komponenten, Debugging, Unit Tests	Mulugeta	Alle
Dokumentation (Code & Projektbericht)	Doxygen-Kommentare, Projektdokumentation, schwierige Code-Stellen	Artian	Alle (Review)
Optional: GUI-Entwicklung	Implementierung einer grafischen Benutzeroberfläche	Nurettin	Optional

Zeitplan (06.05. – 17.06.)

Zeitraum	Arbeitspakete	Meilensteine
06.05 – 12.05	Architektur, Setup	Meilenstein 1: Architektur & Konzept abgeschlossen
13.05 – 19.05	UI, Netzwerk-Kommunikation	Meilenstein 2: Nachrichtenübertragung über CLI möglich
20.05 – 26.05	Discovery-Dienst, IPC, Konfig	Meilenstein 3: Discovery-Dienst arbeitet stabil
27.05 – 09.06	Integration, Dokumentation	Meilenstein 4: System integriert & getestet
10.06 – 14.06	Finalisierung, Puffer, Review	Projektabschluss zur Vorbereitung der Präsentation
17.06	—	Meilenstein 5: Projektpräsentation



Meilenstein

- **Meilenstein 1:** Architektur & Konzept abgeschlossen (Ende Woche 1)
- **Meilenstein 2:** Nachrichtenübertragung über CLI möglich (Ende Woche 2)
- **Meilenstein 3:** Discovery-Dienst arbeitet stabil (Ende Woche 3)
- **Meilenstein 4:** System vollständig integriert & getestet (bis 09.06)
- **Meilenstein 5:** Projektpräsentation (17.06)

Anmerkung

Wir haben uns für Python entschieden, da die Sprache eine gute Balance zwischen Einfachheit und Funktionsumfang bietet. Trotz begrenzter Vorkenntnisse sehen wir dies als Gelegenheit, praxisnah Erfahrung zu sammeln.

Mohammad Milad Inal 1532558

Noel-Leon Hildenbrand 1570029

Mulugeta Gebregergis 1574762

Nurettin Tasoluk 1522479

Artian Mehmeti 1574829