BSRN Chat-System Architekturübersicht

Stand: 22.06.2025

Team: Nabil, Aset, Effe, Daniyal

Inhaltsverzeichnis

Einführung

Diese Dokumentation liefert eine verständliche Übersicht zur Architektur des dezentralen BSRN Chat-Systems. Ziel ist es, die Struktur und die Interaktionen zwischen den Modulen klar zu erklären, sodass auch Nutzer ohne technische Vorkenntnisse den Aufbau des Systems nachvollziehen können.

Konzept & Systemaufbau

Das System basiert auf einem modularen Ansatz mit isolierten Prozessen, die über clevere Mechanismen miteinander kommunizieren. Ein zentraler Steuerprozess lädt die Einstellungen und startet eigenständig die Benutzeroberfläche, das Kommunikationsmodul und den Discovery-Dienst. Jeder dieser Prozesse übernimmt spezifische Aufgaben, was die Wartung erleichtert und die Stabilität erhöht.

Ablauf einer Chat-Interaktion

Beim Versenden einer Nachricht wandelt die Benutzeroberfläche die Eingabe in ein präzises Kommando um und übergibt es an das Kommunikationsmodul. Dieses stellt eine TCP-Verbindung zu einem anderen Client her, überträgt die Nachricht und wartet auf eine Empfangsbestätigung. Empfangene Nachrichten werden durch das Modul an die Oberfläche weitergeleitet und dem Nutzer angezeigt.

Entscheidungskriterien

Wichtige Architekturentscheidungen betreffen die Wahl von UDP für Broadcast-basierte Peer-Erkennung und TCP für zuverlässige Datenübertragung. Hinzu kommt die Nutzung von multiprocessing-Queues zur internen Nachrichtenvermittlung, um Blockierungen zu vermeiden und eine klare Prozessgrenze zu ziehen.

Herausforderungen & Lösungswege

Zu den größten Herausforderungen gehörten das Verhindern von UDP-Broadcast-Überlast und das Verwalten mehrerer gleichzeitiger Verbindungen. Durch den Einsatz zeitversetzter Abfragen und dedizierter Threads konnten diese Probleme effektiv gelöst werden.

Komponenten & Klassen im Überblick

ChatClientUI (Benutzeroberfläche)

Die Klasse ChatClientUI ist für die Interaktion mit dem Nutzer verantwortlich. Sie startet im Hauptprozess und wartet in einer Endlosschleife auf Eingaben, die über die Konsole eingegeben werden. Befehle wie '/msg', '/who' oder '/config' werden durch den integrierten Parser aufbereitet und als Tupel in die 'ui_to_net' Queue geschrieben.

Eingehende Nachrichten liest ChatClientUI aus der 'net_to_ui' Queue aus und zeigt sie formatiert im Terminal an.

Netzwerkmodul

Die Network Klasse läuft in einem separaten Prozess und übernimmt alle Netzwerkoperationen. Sie lauscht zunächst auf Befehle aus der 'ui_to_net' Queue, parst diese und führt sie aus: Bei Textnachrichten öffnet sie mittels Python-Socket eine TCP-Verbindung zum Ziel und sendet das SLCP-Frame. Empfangene Daten von anderen Clients nimmt sie über den TCP-Server-Socket entgegen und legt sie in die 'net_to_ui' Queue, damit die UI sie präsentieren kann.

Discovery-Dienst

Discovery ist ebenfalls ein separater Prozess. Er öffnet einen UDP-Socket im Broadcast-Modus auf dem konfigurierten 'whoisport'. In regelmäßigen Abständen sendet er WHO-Befehle und registriert sich über JOIN. Eingehende JOIN-, LEAVE- und WHO-Frames parst er und aktualisiert ein internes Dictionary mit Peers und Zeitstempeln. Inaktive Peers entfernt ein Cleanup-Thread nach konfigurierbarer Inaktivitätsdauer.

Start-Script

Der StartupCoordinator ist der Einstiegspunkt in main.py. Er initialisiert zuerst alle Queues und startet in der richtigen Reihenfolge DiscoveryService, NetworkHandler und ChatClientUI. Anschließend überwacht er die Prozesse und sorgt bei Programmende für das Senden eines LEAVE-Befehls und ein kontrolliertes Beenden aller Subprozesse.

Einstellungen via config.toml

Die zentrale Konfigurationsdatei config.toml enthält Parameter wie Benutzernamen, Ports für TCP und UDP, Abwesenheitsnachrichten und Pfade für empfangene Bilder. Dies ermöglicht eine einfache Anpassung vor dem Programmstart.

Schritt-für-Schritt Anleitung

1. Installieren Sie Python 3.8+ und pip.
2. Installieren Sie das Modul 'toml' mit pip install toml.
3. Passen Sie config.toml mit Ihrem Handle und Ports an.
4. Starten Sie das System mit python main.py.
5. Geben Sie Ihren Benutzernamen ein und nutzen Sie Befehle wie /help, /who, /msg, /quit.