P2PChat

1.0

Generated by Doxygen 1.9.8

Chapter 1

SLCP Chat-System - Technische Dokumentation

1.1 Einführung

Das **Simple Local Chat Protocol (SLCP) Chat-System** ist eine dezentrale Peer-to-Peer-Anwendung für lokale Netzwerke. Es ermöglicht die direkte Kommunikation zwischen Clients ohne zentrale Server-Infrastruktur.

1.2 Hauptfunktionen

- **Dezentrale Architektur**: Peer-to-Peer-Kommunikation ohne Server
- **Textnachrichten**: Echtzeit-Nachrichtenaustausch
- **Bildübertragung**: Versendung von Bilddateien
- **Automatische Nutzerermittlung**: Discovery-Service für Teilnehmer
- **Abwesenheitsmodus**: Automatische Antworten (Autoreply)
- **Konfigurierbar**: TOML-basierte Konfigurationsdatei
- **Plattformunabhängig**: Python-basierte Implementierung

1.3 Systemarchitektur

Das System basiert auf einer **3-Schichten-Architektur**:

/**

1.3.1 Komponenten

- **Hauptmodul (main.py)** - Koordiniert alle Komponenten
- **Messenger (messenger.py)** - Implementiert SLCP-Protokoll

- **Discovery Service (discovery.py)** - Teilnehmererkennung
- **Konfiguration (config_manager.py)** - TOML-Verwaltung
- **CLI Interface (cli.py)** - Benutzerinteraktion

1.4 SLCP-Protokoll

Das **Simple Local Chat Protocol** definiert:

Befehl	Funktion	Beispiel
JOIN	Anmeldung	`JOIN Alice 5000`
LEAVE	Abmeldung	`LEAVE Alice`
WHO	Teilnehmersuche	`WHO`
KNOWUSERS	Teilnehmerantwort	`KNOWUSERS Alice 192.168.1.42 5000`
MSG	Textnachricht	`MSG Bob "Hallo Welt"`
IMG	Bildnachricht	`IMG Bob 1024`

1.5 Schnellstart

1. **Konfiguration**: Bearbeite `config.toml`
2. **Start**: `python3 main.py`
3. **Teilnehmer finden**: `who`
4. **Nachricht senden**: `send <handle> <nachricht>`
5. **Bild senden**: `img <handle> <pfad>`

1.6 Installation

1.7 Entwicklung

- **Sprache**: Python 3.8+
- **Protokoll**: UDP/TCP basiert
- **IPC**: Multiprocessing Queues
- **Konfiguration**: TOML-Format
- **Dokumentation**: Doxygen + Graphviz

1.8 Lizenz

Author

Noah Wolde Suphi Dogruel, Sebastian Stautz , Issa Waheed, Khanh-Tam Vu

Date

22.06.2025 Sommersemester 2025

Version

1.0

See also

Systemarchitektur

SLCP-Protokoll

Bedienungsanleitung

Chapter 2

BSRN Projekt – Peer-to-Peer Chat (SLCP)

Ein dezentrales Chat-System mit Text-/Bildunterstützung, entwickelt für das Modul "Betriebssysteme und Rechnernetze" an der UAS.

2.1 Installation

```
# Voraussetzungen
sudo apt install python3 python3-pip

# Repository klonen
git clone https://github.com/Woah-Nolde/P2PChat
cd p2p-chat

# Abhängigkeiten installieren
pip install -r requirements.txt
```

2.2 Konfiguration

Bearbeiten Sie `config.toml`:

```
handle = "Ihr_Name"
port = 5000          # Client-zu-Client-Port
whoisport = 4000     # Discovery-Broadcast-Port
autoreply = "Abwesend"
imagepath = "./received_images"
```

2.3 Systemarchitektur

```
graph TD
    A[main.py\nCLI-UI] <-->|IPC/Queues| B[messenger.py\nNetzwerk]
    B <-->|UDP Broadcast/IPC| C[discovery.py]
    B -->|Unicast| D[Andere Clients]
    A <-->|read/write| E[config.toml]
```

Prozessaufteilung:

Prozess	Verantwortlichkeit	Protokolle
`main.py`	Nutzerinteraktion	IPC/Broadcast (Sockets/Queues)
`messenger.py`	Nachrichtenaustausch	UDP/IPC Unicast/Broadcast
`discovery.py`	Teilnehmererkennung	UDP/IPC Broadcast

2.4 Bedienung

Starten:

```
python3 main.py
```

CLI-Befehle:

Befehl	Aktion	Beispiel
`/who`	Aktive Nutzer anzeigen	`/Entdeckte Nutzer: Alice`
`/send <user> <text>`	Nachricht senden	`/send Bob Hallo!`
`/img <user> <path>`	Bild senden	`/img Bob ~/pic.jpg`
`/quit`	Chat verlassen	`/Alice hat den Chat verlassen`
`/abwesend`	[Abwesend-Modus]	`/Abwesend-Modus`

2.5 Protokoll (SLCP)

Befehl	Format	Beispiel
`JOIN`	`JOIN <handle> <port>`	`JOIN Alice 5000`
`LEAVE`	`LEAVE <handle>`	`LEAVE Alice`
`MSG`	`MSG <handle> <text>`	`MSG Bob "Hallo"`
`IMG`	`IMG <handle> <size>` + Binärdaten	`IMG Bob 2048`

2.6 Dokumentation

- Code-Dokumentation: Generieren mit `doxygen Doxyfile`
- Protokollspezifikation: Siehe Projektunterlagen

2.7 Team

Name	Rolle	Komponente
Team-Mitglied 1	Netzwerkschicht	messenger.py
Team-Mitglied 2	Discovery-Dienst	discovery.py
Team-Mitglied 3	Benutzeroberfläche	main.py

2.8 Logo

2.9 Screenshot der CLI

Lizenz: MIT – Frankfurt University of Applied Sciences, 2025 ```

Chapter 3

Bedienungsanleitung

3.1 Installation

3.1.1 Voraussetzungen

Stellen Sie sicher, dass die folgenden Pakete installiert sind:

```
sudo apt update
sudo apt install python3 python3-pip
```

3.1.2 Projekt klonen und einrichten

Klonen Sie das Repository und installieren Sie die Abhängigkeiten:

```
git clone https://github.com/Woah-Nolde/P2PChat
cd p2p-chat
pip install -r requirements.txt
```

3.2 Konfiguration

3.2.1 Konfigurationsdatei bearbeiten

Die Konfiguration erfolgt über die Datei `config.toml`. Bearbeiten Sie diese gemäß Ihren Bedürfnissen:

```
[user]
handle = "Ihr_Benutzername"
autoreply = "Ich bin gerade nicht erreichbar."

[network]
port_range = [5000, 5100]  # Port-Bereich für UDP-Kommunikation
whoisport = 4000           # Port für Discovery-Broadcasts

[files]
imagepath = "./src/image"  # Pfad für gespeicherte Bilder
```

3.2.2 Konfiguration über CLI

Das Programm bietet auch eine interaktive Konfiguration:

- `config show` - Aktuelle Konfiguration anzeigen
- `config edit` - Interaktive Konfigurationsbearbeitung
- `config reload` - Konfiguration neu laden

3.3 Nutzung

3.3.1 Programmstart

Starten Sie das Programm mit:

```
cd src
python3 main.py
```

3.3.2 Befehlsreferenz (CLI)

3.3.2.1 Discovery und Nutzer

- **who:** Liste aller aktiven Nutzer im Netzwerk suchen
- **users:** Bereits bekannte Nutzer anzeigen

3.3.2.2 Kommunikation

- **send** <handle> <nachricht>: Textnachricht an Nutzer senden
  ```
  send Alice Hallo, wie geht es dir?
  ```
- **img** <handle> <pfad_zum_bild>: Bild an Nutzer senden
  ```
  img Bob ./bilder/urlaubsfoto.jpg
  ```

3.3.2.3 Status und Verwaltung

- **abwesend:** Abwesenheitsmodus aktivieren (Autoreply)
- **name:** Handle (Benutzername) ändern
- **quit:** Chat verlassen und Programm beenden

3.3.2.4 Konfiguration

- **config:** Konfigurationshilfe anzeigen
- **config** show: Aktuelle Konfiguration anzeigen
- **config** edit: Interaktive Konfigurationsbearbeitung
- **config** reload: Konfiguration neu laden

3.4 Spezielle Funktionen

3.4.1 Abwesenheitsmodus (Autoreply)

Der Abwesenheitsmodus ermöglicht automatische Antworten:

1. Aktivierung mit dem Befehl `abwesend`
2. Automatische Antwort mit konfigurierter Nachricht
3. Deaktivierung durch Drücken der Enter-Taste

3.4.2 Handle-Kollisionen

Bei doppelten Handles wird automatisch eine Nummer angehängt:

- Alice → Alice2 (bei Kollision)
- Alice2 → Alice3 (bei weiterer Kollision)

3.4.3 Bildübertragung

- Maximale Bildgröße: 50 KB
- Unterstützte Formate: Alle binären Formate
- Empfangene Bilder werden in `./src/image/` gespeichert
- Dateinamensformat: `empfangen_<handle>_<timestamp>.jpg`

3.5 SLCP-Protokoll

3.5.1 Nachrichtenformat

Das Simple Local Chat Protocol (SLCP) verwendet folgende Formate:

3.5.1.1 Discovery-Nachrichten

- **JOIN** <Handle> <Port>: Beim Netzwerk anmelden
- **LEAVE** <Handle>: Netzwerk verlassen
- **WHO:** Aktive Nutzer abfragen
- **KNOWUSERS** <Liste>: Antwort auf WHO-Anfrage

3.5.1.2 Chat-Nachrichten

- **MSG** <Handle> <Nachricht>: Textnachricht
- **IMG** <Handle> <Größe>: Bild-Header vor Binärdaten

3.6 Netzwerk-Architektur

3.6.1 Verwendete Ports

- **4000**: Discovery-Service (JOIN/LEAVE/WHO)
- **4001**: Discovery-Events (USERJOIN/USERLEAVE broadcasts)
- **5000-5100**: Dynamische Portzuweisung für Chat-Kommunikation

3.6.2 Discovery-Dienst

- Automatischer Start beim ersten Programmstart
- Singleton-Pattern verhindert mehrfache Instanzen
- Broadcast-basierte Nutzer-Erkennung im LAN

3.7 Problembehandlung

3.7.1 Häufige Probleme

3.7.1.1 Port bereits belegt

```
[Discovery] Port 4000 bereits belegt - Discovery läuft bereits
```

Lösung:∗∗ Discovery-Dienst läuft bereits. Normaler Betrieb.

3.7.1.2 Kein freier Port

```
Kein freier UDP-Port im Bereich 5000-5010 gefunden!
```

Lösung:∗∗ Port-Bereich in `config.toml` erweitern.

3.7.1.3 Bild zu groß

```
[Fehler] Bild zu groß (75000 Bytes). Maximal erlaubt: 512 Bytes.
```

Lösung:∗∗ Bild verkleinern oder komprimieren.

3.7.2 Debug-Informationen

Das Programm gibt detaillierte Statusmeldungen aus:

- `[Discovery]`: Discovery-Service-Meldungen
- `[Nachricht]`: Eingehende Textnachrichten
- `[Sender]/[Empfänger]`: Bildübertragung
- `[Fehler]`: Fehlermeldungen

3.8 Erweiterte Nutzung

3.8.1 Mehrere Instanzen

Für Tests können mehrere Instanzen mit unterschiedlichen Handles gestartet werden:

```
# Terminal 1
python3 main.py  # Alice

# Terminal 2 (anderen Handle in config.toml setzen)
python3 main.py  # Bob
```

3.8.2 Netzwerk-Diagnose

Zur Diagnose von Verbindungsproblemen:

1. `who` - Aktive Nutzer suchen
2. `users` - Bekannte Nutzer auflisten
3. Ping-Test zu anderen Rechnern im LAN

3.9 Wichtige Hinweise

- Das System arbeitet nur im lokalen Netzwerk (LAN)
- IPv6 wird teilweise unterstützt, IPv4 wird bevorzugt
- Nachrichten sind auf 512 Zeichen begrenzt
- Der Discovery-Service muss nur einmal pro Netzwerk laufen
- Empfangene Bilder werden automatisch gespeichert
- Bei Programmabbruch (Ctrl+C) wird automatisch LEAVE gesendet

3.10 Siehe auch

- Projektarchitektur
- README-Datei
- SLCP-Protokollspezifikation

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 6

Class Index

6.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 7

File Index

7.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 8

Namespace Documentation

8.1 src.discovery Namespace Reference

Functions

- ensure_singleton (port, disc_to_ui)
- discoveryloop (net_to_disc, disc_to_net, disc_to_ui, DISCOVERY_PORT)

8.1.1 Detailed Description

```
@file discovery.py
@brief Implementierung des Discovery-Servers für das SLCP-Chat-Programm
@details Dieser Dienst verwaltet die Teilnehmerliste und antwortet auf Broadcast-Anfragen.
@date 21.05.2025
```

8.1.2 Function Documentation

8.1.2.1 discoveryloop()

```
src.discovery.discoveryloop (
              net_to_disc,
              disc_to_net,
              disc_to_ui,
              DISCOVERY_PORT )

@brief Hauptfunktion des Discovery-Dienstes
@param net_to_disc IPC-Queue für Nachrichten vom Netzwerkmodul
@param disc_to_net IPC-Queue für Nachrichten an das Netzwerkmodul
@param disc_to_ui IPC-Queue für Nachrichten an die Benutzeroberfläche
@param DISCOVERY_PORT Port für den Discovery-Dienst
@details Diese Funktion implementiert den Hauptloop des Discovery-Servers:
         - Verwaltet die Liste aktiver Teilnehmer
         - Verarbeitet JOIN/LEAVE/WHO Nachrichten
         - Sendet Antworten auf WHO-Anfragen
```

8.1.2.2 ensure_singleton()

```
src.discovery.ensure_singleton (
              port,
              disc_to_ui )

@brief Stellt sicher, dass nur eine Instanz des Discovery-Dienstes läuft
@param port Der Port, auf dem der Dienst laufen soll
@param disc_to_ui IPC-Queue für Nachrichten an die Benutzeroberfläche
@details Versucht, den angegebenen Port zu binden. Falls dies fehlschlägt,
         wird angenommen, dass bereits eine Instanz läuft und das Programm beendet.
```

Chapter 9

Class Documentation

9.1 <JOIN> Protocol Reference

Sendet JOIN-Broadcastnachricht <Port>
Collaboration diagram for <JOIN>:

9.1.1 Detailed Description

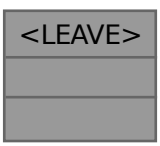
Sendet JOIN-Broadcastnachricht <Port>
The documentation for this protocol was generated from the following file:

- src/main.py

9.2 <LEAVE> Protocol Reference

Sendet LEAVE-Nachricht an alle

Collaboration diagram for <LEAVE>:



9.2.1 Detailed Description

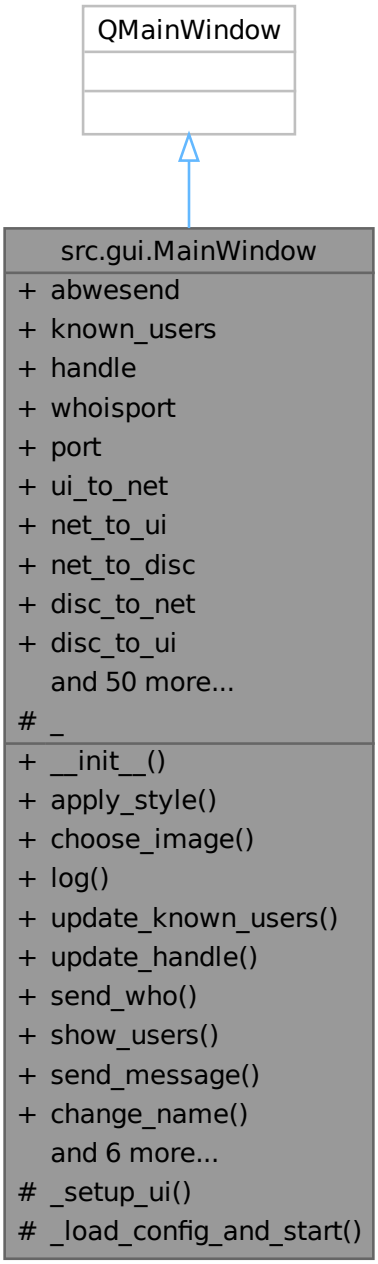
Sendet LEAVE-Nachricht an alle

The documentation for this protocol was generated from the following file:

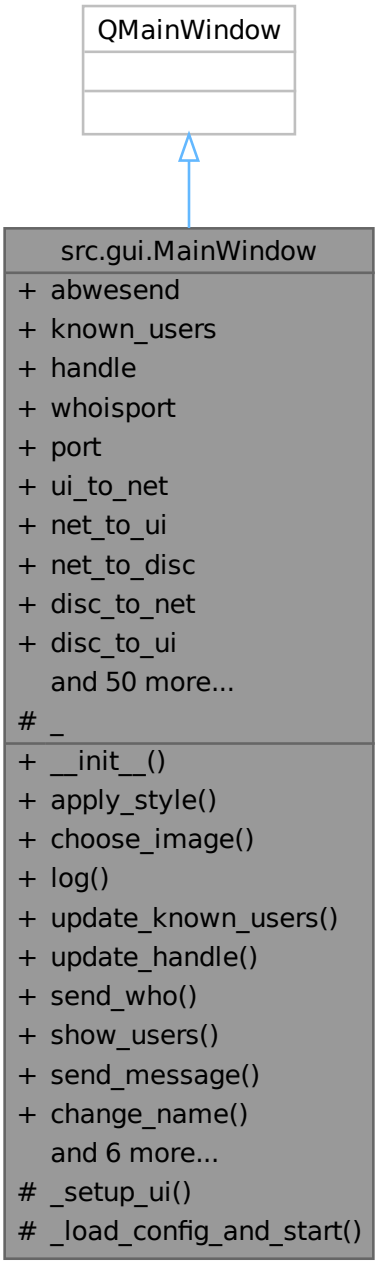
- src/main.py

9.3 src.gui.MainWindow Class Reference

Inheritance diagram for src.gui.MainWindow:



Collaboration diagram for src.gui.MainWindow:



Public Member Functions

- **__init__** (self)
- **apply_style** (self)
- **choose_image** (self)
- **log** (self, text)
- **update_known_users** (self, users)
- **update_handle** (self, new_handle)

- **send_who** (self)
- **show_users** (self)
- **send_message** (self)
- **change_name** (self)
- **edit_config_popup** (self)
- **save** ()
- **edit_config** (self)
- **reload_config** (self)
- **toggle_abwesend** (self)
- **quit_chat** (self)

Static Public Attributes

- **abwesend**
- **known_users**
- **handle**
- **whoisport**
- **port**
- **ui_to_net**
- **net_to_ui**
- **net_to_disc**
- **disc_to_net**
- **disc_to_ui**
- **network_process**
- **discovery_process**
- **reader_thread**
- str **style**
- **options** = QFileDialog.Options()
- **file_path**
- **central_widget** = QWidget()
- **vbox** = QVBoxLayout()
- **chat_log**
- **h_users** = QHBoxLayout()
- **user_list_widget**
- **h_input** = QHBoxLayout()
- **input_handle**
- **input_message**
- **send_msg_btn**
- **send_img_btn**
- **h_buttons** = QHBoxLayout()
- **btn_who**
- **btn_users**
- **btn_name**
- **btn_config**
- **btn_reload**
- **btn_abwesend**
- **btn_quit**
- **config** = load_config()
- **port_range** = config["network"]["port_range"]
- **target** = self.input_handle.text().strip()
- **network_main**
- **args**
- **discoveryloop**
- **daemon**
- str **users_str** = ", ".join(self.known_users.keys())

- **text** = self.input_message.text().strip()
- **ip**
- tuple **image_extensions** = ('.png', '.jpg', '.jpeg', '.bmp', '.gif')
- int **MAX_IMAGE_SIZE** = 50 ∗ 1024
- **size** = os.path.getsize(text)
- **pixmap** = QPixmap(text)
- **scaled** = pixmap.scaledToWidth(200)
- **cursor** = self.chat_log.textCursor()
- **b64_data** = base64.b64encode(img_file.read()).decode('utf-8')
- **new_handle**
- **ok**
- **dlg** = QDialog(self)
- **layout** = QFormLayout()
- **handle_input** = QLineEdit(config["user"].get("handle", ""))
- **autoreply_input** = QLineEdit(config["user"].get("autoreply", ""))
- **port_input** = QSpinBox()
- **discovery_input** = QLineEdit(config["network"].get("discovery", ""))
- **button_box** = QDialogButtonBox(QDialogButtonBox.Save | QDialogButtonBox.Cancel)

Protected Member Functions

- **_setup_ui** (self)
- **_load_config_and_start** (self)

Static Protected Attributes

- _

The documentation for this class was generated from the following file:

- src/gui.py

9.4 src.gui.ReaderThread Class Reference

Inheritance diagram for src.gui.ReaderThread:

Collaboration diagram for src.gui.ReaderThread:

Public Member Functions

- **__init__** (self, disc_to_ui, net_to_ui, ui_to_net, handle, port)
- **run** (self)

Public Attributes

- **disc_to_ui**
- **net_to_ui**
- **ui_to_net**
- **handle**
- **port**
- **abwesend**
- **known_users**

Static Public Attributes

- **new_log** = pyqtSignal(str)
- **update_users** = pyqtSignal(dict)
- **update_handle** = pyqtSignal(str)
- **abwesend_mode** = pyqtSignal(bool)

The documentation for this class was generated from the following file:

- src/gui.py

Chapter 10

File Documentation

10.1 src/config_manager.py File Reference

Verwaltet die TOML-Konfigurationsdatei.

Functions

- **src.config_manager.load_config** (path=conf_file)
  Lädt die Konfiguration aus einer TOML-Datei.
- **src.config_manager.save_config** (config, path=conf_file)
  Speichert die Konfiguration in eine TOML-Datei.
- **src.config_manager.show_config** (config)
  Zeigt die aktuelle Konfiguration an.
- **src.config_manager.parse_toml_type** (value)
  Konvertiert TOML-Werte zu Python-Typen.
- **src.config_manager.edit_config** ()
  Konfigurationsbearbeitung
- **src.config_manager.lookup_handle** (handle, config=None)
  Sucht einen Nutzer in known_users.
- **src.config_manager.save_image** (handle, data)
  Speichert ein empfangenes Bild.
- **src.config_manager.handle_autoreply** (sender_ip, sender_port, config)
  Sendet eine Autoreply-Nachricht an den Absender.

Variables

- dict **src.config_manager.known_users** = {}
- str **src.config_manager.conf_file** = "config/config.toml"

10.1.1 Detailed Description

Verwaltet die TOML-Konfigurationsdatei.

10.2 src/main.py File Reference

Hauptmodul des P2P-Chat-Systems.

Functions

- **src.main.print_prompt** ()
- **src.main.get_own_ip** ()

  Gibt die eigene LAN-IP-Adresse zurück.
- **src.main.show_net_and_disc_messages** (disc_to_ui, net_to_ui, my_handle, my_port, ui_to_net)

  Zeigt Netzwerk- und Discovery-Nachrichten an.
- **src.main.send_join** (handle, port)
- **src.main.send_leave** (handle, whoisport, known_users)
- **src.main.cli_loop** (whoisport, ui_to_net, net_to_ui, port, p1, p2)

  Haupt-CLI-Loop
- **src.main.find_free_port** (start_port, end_port)

  Findet freien UDP-Port.
- src.main.main ()

  Hauptfunktion.

10.2.1 Detailed Description

Hauptmodul des P2P-Chat-Systems.
Koordiniert alle Komponenten (CLI, Netzwerk, Discovery) und startet die Prozesse

10.2.2 Function Documentation

10.2.2.1 main()

```
src.main.main ( )
```

Hauptfunktion.
Startet:

1. Konfiguration
2. Netzwerk-Prozess
3. Discovery-Prozess
4. CLI-Oberfläche

Index