TW Mailer Pro - Protokoll

Im Vergleich zur ursprünglichen Basic-Variante des TWMailers haben wir wesentliche Erweiterungen und Anpassungen vorgenommen, um die Funktionalität des Programms zu verbessern.

1. Client- und Server-Architektur

- **Client**:
  In der Basic-Version unterstützte unser Client Befehle wie SEND, LIST, READ, DEL und QUIT. In der erweiterten Version haben wir diese Befehle ergänzt:
  - **REGISTER** und **LOGIN**: Damit können Benutzer sich registrieren und authentifizieren.
  - **LOGOUT**: Um die Sitzung eines Benutzers sicher zu beenden.
  - **LIST unread**: Für eine bessere Übersicht über ungelesene Nachrichten.
  - **UPDATE**: Um den Lesestatus zwischen mehreren verbundenen Clients synchron zu halten.

Zusätzlich kann unser Client jetzt fragmentierte Nachrichten senden und empfangen, was den Umgang mit großen Datenmengen ermöglicht.

- **Server**:
  Unser Server war in der Basic-Version auf die Speicherung von Nachrichten in einer einfachen Dateistruktur ausgelegt. Für die erweiterte Version haben wir:
  - **Benutzerverwaltung** hinzugefügt, um Benutzerdaten sicher zu speichern und Session-Informationen zu verwalten.
  - **Authentifizierung** eingeführt, damit nur registrierte und angemeldete Benutzer Zugriff auf ihre Nachrichten haben.
  - **Nachrichtenfilter** implementiert, die es ermöglichen, Nachrichten basierend auf Attributen zu filtern, z. B. ungelesene Nachrichten anzuzeigen.
  - **Multi-Threading**, um parallele Verbindungen besser zu verwalten und die Server-Performance zu steigern.

2. Verwendete Technologien und Bibliotheken

Wir haben die grundlegenden Technologien der Basic-Version beibehalten, aber einige Erweiterungen vorgenommen:

- **Sicherheit**:
  - In der Basic-Version gab es keine Sicherheitsmaßnahmen. Jetzt verschlüsseln wir die gesamte Kommunikation zwischen Client und Server mit **OpenSSL**.
  - Passwörter speichern wir nicht mehr im Klartext, sondern sicher gehasht mit **bcrypt**.
- **Threading und Synchronisation**:
  - Während wir in der Basic-Version bereits pthread nutzten, haben wir in der erweiterten Version auf **std::mutex** umgestellt, um eine feinere Kontrolle über parallele Prozesse zu haben.

3. Entwicklungsstrategie und notwendige Protokollanpassungen

- **Planung und Definition**:
  - Neue Befehle wie **REGISTER** und **LOGIN** für Benutzerverwaltung
  - Bestehende Befehle wie **LIST** haben wir erweitert, z. B. mit der Filterfunktion (**LIST unread**).
- **Implementierung neuer Funktionen**:
  - **Sitzungsmanagement**: Wir haben Session-Tokens eingeführt, die bei jedem Befehl eines Clients geprüft werden.
  - **Fehlerbehandlung**: Unser Server reagiert jetzt mit klar definierten Fehlermeldungen wie ERROR InvalidSession, wenn z. B. eine Sitzung abgelaufen ist.

4. Verwendete Synchronisationsmethoden

- **Mutexes und Locks**:
  - Mit **std::mutex** steuern wir jetzt den Zugriff auf gemeinsam genutzte Ressourcen wie Benutzer- oder Nachrichtendaten effizienter.
- **Lesestatus-Synchronisation**:
  - Mit dem neuen **UPDATE**-Befehl ermöglichen wir es Benutzern, den Lesestatus ihrer Nachrichten zwischen mehreren Clients in Echtzeit abzugleichen.

5. Umgang mit großen Nachrichten

- **Fragmentierung**:
  - Wir haben Nachrichten in Blöcke aufgeteilt, die schrittweise übertragen werden. Der Server setzt diese Blöcke anhand eines eindeutigen Identifikators wieder zusammen.
- **Protokollerweiterung**:
  - Der SEND-Befehl wurde durch SEND_PART ergänzt, um die Fragmentierung zu unterstützen, z.B. SEND_PART <message_id> <chunk_number> <data>.
- **Speicheroptimierung**:
  - Mit dynamischem Puffermanagement verhindern wir, dass der Server durch große Nachrichten überlastet wird.