Hochschule Kaiserslautern
University of Applied Sciences

Informatik und Mikrosystemtechnik
Zweibrücken

Entwicklung verteilter Anwendungen „Schiffe versenken”

Maurice Didion Julian Paulus Alexander Liggesmeyer

04.02.2020

Zusammenfassung

Dieser Artikel dient als Projektdokumentation zu dem Projekt „Schiffe versenken”, welches im Rahmen der Vorlesung „Entwicklung verteilter Anwendungen” an der Hochschule Kaiserslautern entwickelt wurde. Dieser Artikel befasst sich sowohl mit der Bedienung der grafischen Benutzeroberfläche, sowie mit dem Spielablauf und der Technik, die hinter dem Spiel steckt. Des Weiteren wird verdeutlicht, wie die Kommunikation zwischen dem Server und mehreren Clienten realisiert wurde.

Inhaltsverzeichnis

1 Einleitung

Dieser Projektbericht handelt von dem Spiel ,,Schiffe versenken", welches im Rahmen der Vorlesung ,,Entwicklung verteilter Anwendungen" an der Hochschule Kaiserslautern entwickelt wurde. Das Spiel besteht aus zwei Programmen, welche beide mit der objektorientierten Programmiersprache ,,Java" umgesetzt wurden: Dem Server, welcher Spielerkonten verwaltet, neue Spiele in Form von Lobbies starten kann und mehrere Spiele parallel bearbeiten kann, sowie dem Clienten, welcher eine grafische Benutzeroberfläche (GUI) für die Spieler bereitstellt und mit dem Server kommuniziert. Beide Programme kommunizieren durch einen TCP-Socket miteinander, welches vom Login bis zum Ende des Spieles dauerhaft geöffnet bleibt.

2 Die Spielregeln

Zu Beginn eines Spieles setzt jeder Spieler seine Schiffe auf das 10x10 Felder große Spielfeld. Es gibt insgesamt 5 Schiffe in verschiedenen Größen. Jeweils ein 2, 4 und 5 Felder großes Schiff und zwei 3 Felder große. Schiffe können vertikal und horizontal platziert werden. Zwischen den Schiffen muss immer ein Abstand von mindestens einem Feld in alle Richtungen bestehen. Das Platzieren der Schiffe erfolgt verdeckt. Beide Spieler dürfen die Position der gegnerischen Schiffe nicht wissen. Nachdem beide Spieler ihre Schiffe platziert haben schießt der erste Spieler mit einem Klick auf ein ausgewähltes, gegnerisches Feld. Steht an der ausgewählten Position ein gegnerisches Schiff, wird das Spielfeld rot angezeigt. Schießt er daneben wird das Feld grau. Im Falle eines Treffers darf der Spieler noch einmal schießen. Hat er jedoch daneben geschossen, ist der gegnerische Spieler am Zug. Hat ein Spieler ein Schiff versenkt, wird ihm dies im Chat des Spieles angezeigt. Ziel des Spieles ist es, die Schiffe des jeweils anderen Spielers vollständig zu zerstören.

3 Die Benutzeroberfläche

3.1 Das Login-Fenster

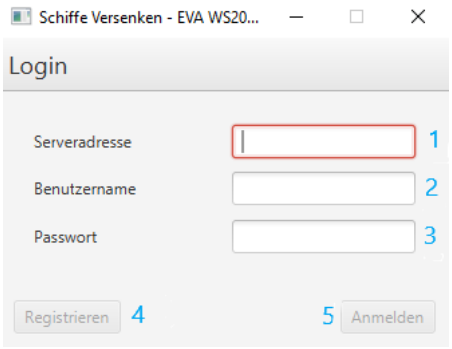

Abbildung 1: Login-Fenster

Das Login-Fenster besitzt drei Textfelder zum Eintragen von Serveradresse (1), Benutzernamen (2) und Passwort (3). Wurden diese Felder befüllt, werden die Buttons zum Anmelden (5) bzw. zum Registrieren (4) freigeschaltet. Wurde der Benutzer neu registriert, so wird er anschließend automatisch eingeloggt.

3.2 Die Lobby-Auswahl

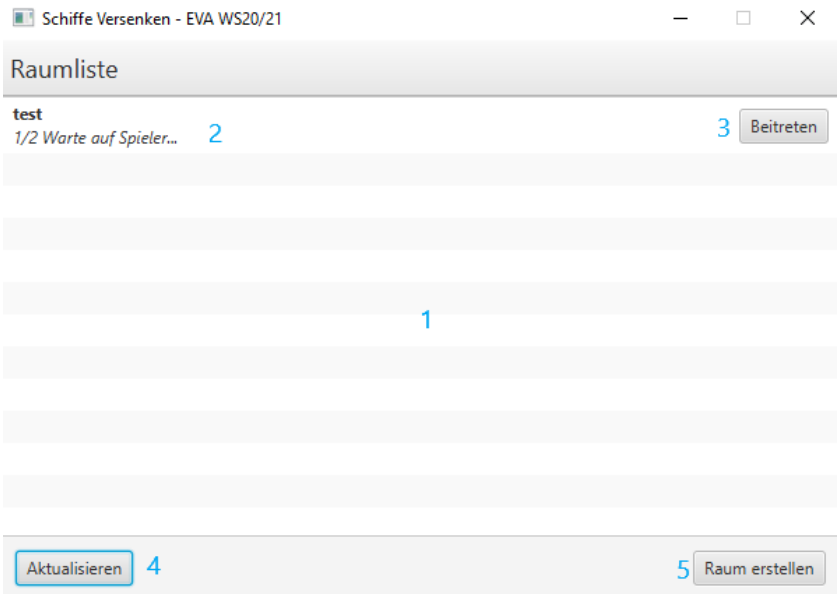

Abbildung 2: Lobby-Fenster

Nach erfolgreichem Login, wird der Nutzer automatisch auf die Lobby-Auswahl weitergeleitet. Diese besteht aus einer ListView (1), welche alle bis zu diesem Zeitpunkt offenen Lobbys (in diesem Fall nur eine) (2) anzeigt. Jede Lobby besitzt einen Button zum Beitreten (3). Des Weiteren gibt es einen Button, welcher es erlaubt eine neue Lobby (ein neues Spiel) zu erstellen (5), sowie einen Button zum Aktualisieren der Liste (4).

3.3 Das Spielfeld

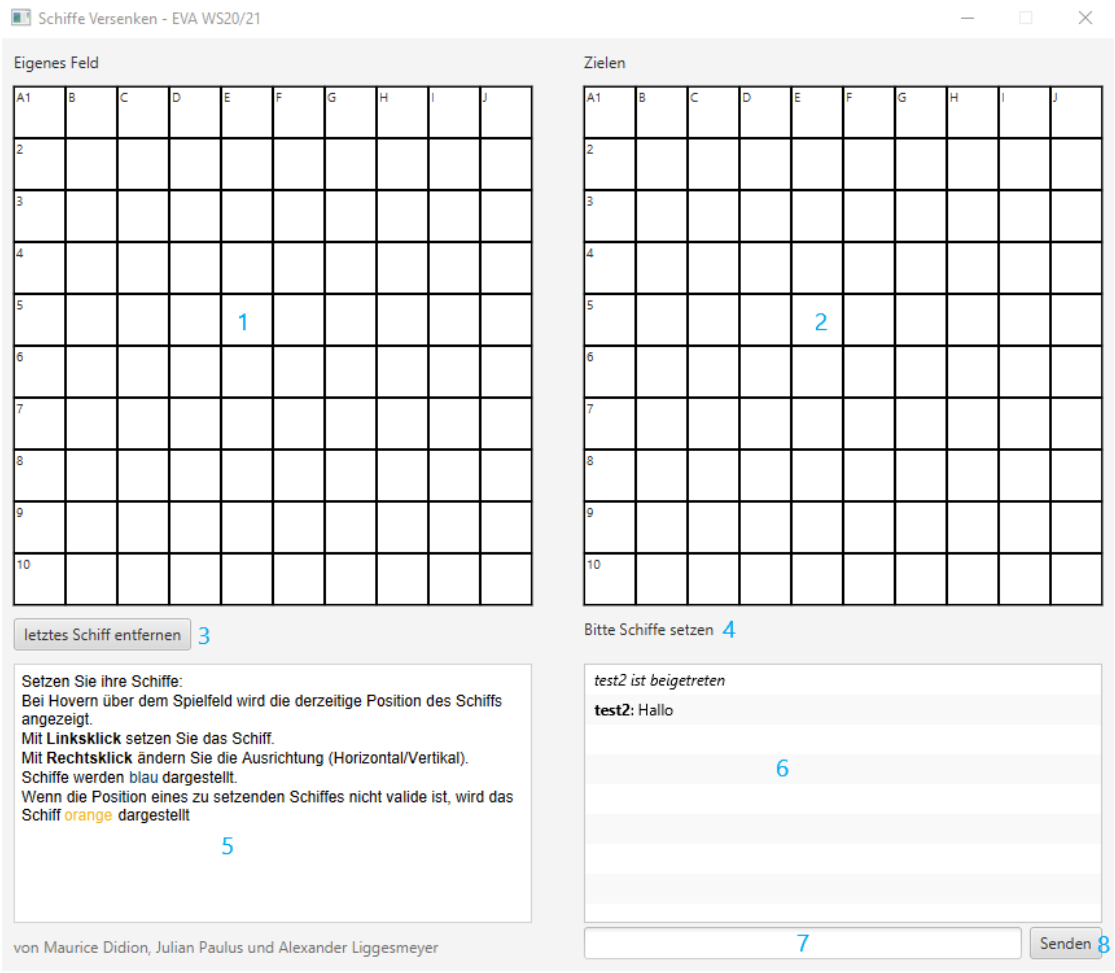

Abbildung 3: Das Spielfeld

Das Spielfeld (Abbildung 3) besteht aus zwei GridPanes, welche mit Labels befüllt wurden und als Feld zum Zielen (2), bzw. als Feld für die Schiffe des Spielers (1) dienen, einem Button (3) zum Entfernen des letzten hinzugefügten Schiffes, der nur während dem Setzen der Schiffe aktiv ist, einem Textfeld, welches die Regeln der aktuellen Runde anzeigt (5), einem Status-Label (4), welches den Status der derzeitigen Runde/des Spieles anzeigt, einer ListView (6) für Chat-Nachrichten und Meldungen, sowie einer Textbox (7) und einem Button (8) zum Senden von Nachrichten. Durch das Betätigen des Senden-Buttons wird die Nachricht an den Server und von dort aus weiter an den anderen Spieler geschickt. Bei Erhalten einer Nachricht, wird diese automatisch in die ListView (6) eingefügt.
Das Status-Label (4) und das Textfeld (5) ändern sich je nach Phase des Spieles automatisch und zeigen dem Spieler nützliche Informationen bezüglich der Spielregeln, der Steuerung und was in der aktuellen Phase zu tun ist an. Des weiteren melden diese am Ende des Spieles ob der Spieler gewonnen, beziehungsweise ob dieser verloren hat.
Die Gridpanes besitzen je nach Rundenstatus unterschiedliche MouseHover-/MouseKlick-Events,

wobei beide inaktiv sind, solange sich nicht zwei Spieler im Spiel befinden. Nach dem Ende des Spieles, wird unterhalb des Spielerfeldes ein Rematch button eingeblendet, welcher es den Spielern erlaubt ein neues Spiel zu starten. Verlässt ein Spieler das Spiel, so wird dieser wieder ausgeblendet.

3.3.1 Das Spielerfeld (1)

Während der ersten Phase, dem Setzen der Schiffe, ist dieses Feld bei beiden Spielern aktiv. Sobald man in dieser Phase mit der Maus über das Feld ,,hovert', wird das aktuell zu platzierende Schiff, sofern ein Platzieren an dieser Stelle möglich ist, angezeigt. Würde das Schiff über den Spielfeldrand hinausgehen, wird dieses ausgeblendet. Würde es zu nahe an einem anderen Schiff, oder sogar auf diesem platziert werden, so wird es gelb eingefärbt. Durch einen Rechtsklick kann die Ausrichtung des Schiffes geändert werden und durch einen Linksklick kann das aktuelle Schiff gesetzt werden. Das Spiel schaltet anschließend automatisch zum nächsten Schiff. Über den Button ,,Letztes Schiff entfernen" (3), kann das zuletzt gesetzte Schiff wieder entfernt werden. Wurden alle Schiffe gesetzt, wird entweder auf den anderen Spieler gewartet, oder es wird gleich in die Spielphase übergegangen.
Außerhalb der Setup-Phase, in der jeder Spieler seine Schiffe setzt, gibt es keine vom Spieler auslösbare Events auf dem Spielerfeld. Sendet der Server ein Paket mit den durch den Gegner beschossenen Koordinaten, wird das entsprechende Feld lediglich eingefärbt: rot für einen Treffer, grau für einen verfehlten Schuss.

3.3.2 Das Zielfeld (2)

Hier kann der Spieler in der Spielphase auf den Gegner feuern. Hierzu ,,hovert" er zunächst über das Zielfeld, wobei die derzeit ausgewählte Koordinate gelb hervorgehoben wird. Durch einen Linksklick wird auf den Gegner geschossen und ein entsprechendes Paket an den Server gesendet, woraufhin dieser mit einem Paket mit den Informationen, ob getroffen wurde oder nicht, antwortet. Entsprechend dieser Antwort wird die Koordinate eingefärbt: rot für einen Treffer und Grau für verfehlt.

4 Die Technik

4.1 Verbindungsmanagement

Die eigentlichen Verbindungen (Sockets) werden mit Hilfe der ,,Connection"-Klasse abstrahiert. Connections können Pakete senden und empfangen, wobei das Empfangen in einem PacketReader-Thread stattfindet, welcher der Connection zugeordnet ist. Auf Serverseite gibt es verschiedene Verbindungstypen, mehr dazu in Abschnitt 4.2.2 ,,Pakete empfangen".

4.1.1 ConnectionService

Der ConnectionService (Abbildung 4) speichert intern alle offenen Verbindungen des Servers. Wenn die Verbindung autorisiert ist, merkt der Service sich außerdem den passenden Spieler zur Verbindung. Der Service existiert als Singleton im Server. Beim Erstellen oder Upgraden einer Verbindung wird die Verbindung initial gespeichert und später gegebenenfalls ausgetauscht. Der ConnectionService beobachtet alle offenen Verbindungen und wird benachrichtigt, falls eine Verbindung geschlossen wird, wodurch sie aus dem Speicher des Services entfernt wird. Der Service wird unter anderem dazu verwendet, um das mehrfache Einloggen des gleichen Spielers zu verhindern (mehr dazu in Abschnitt 4.3 ,,Benutzermanagement").

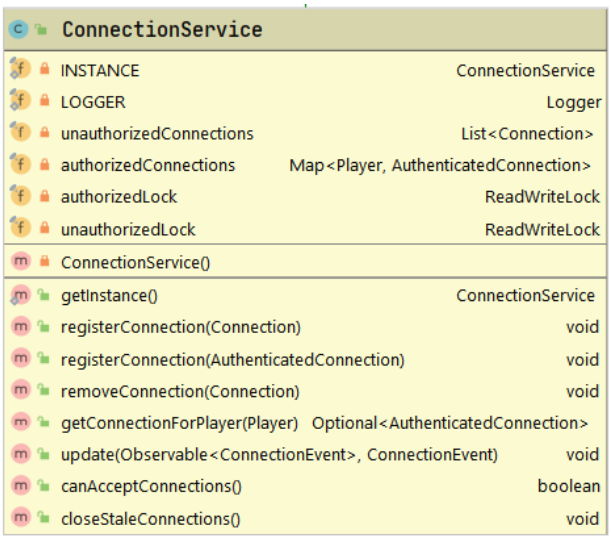

Abbildung 4: ConnectionService Klassendiagramm

4.2 Paketh andling

Um die Kommunikation zwischen Client und Server zu erleichtern, sind alle Pakete nach einem bestimmten Schema aufgebaut (vgl. Abbildung 5). Alle Pakete besitzen einen Identifier. Dieser Identifier ist immer das erste Byte eines Paketes und ermöglicht es dem Empfänger, den für das entsprechende Paket korrekten unmarshalling-Prozess auszuwählen. Die Pakete sind sowohl im Clienten als auch im Server in verschiedenen Ausführungen zu finden. Die jeweils sendende Seite besitzt nur den Code zum Generieren des Paketes und die empfangende Seite den Code zum Empfangen und Verarbeiten.

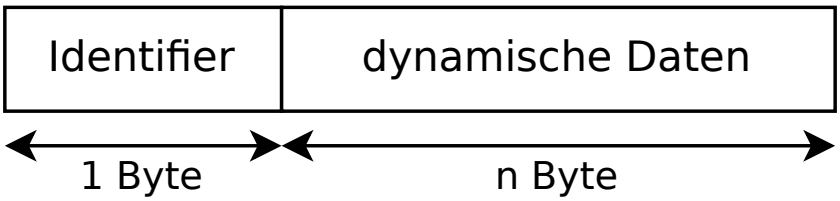

Abbildung 5: Genereller Aufbau eines Paketes (Beispiel für ein genaues Paket: siehe Abbildung 6)

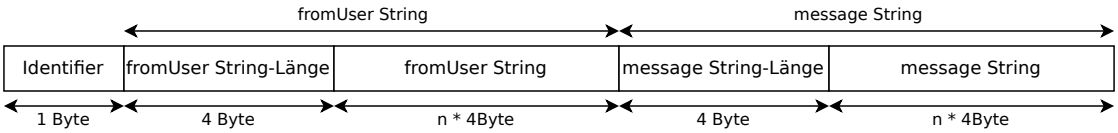

Abbildung 6: Aufbau des ChatMessagePacket-Paketes

4.2.1 Pakete senden

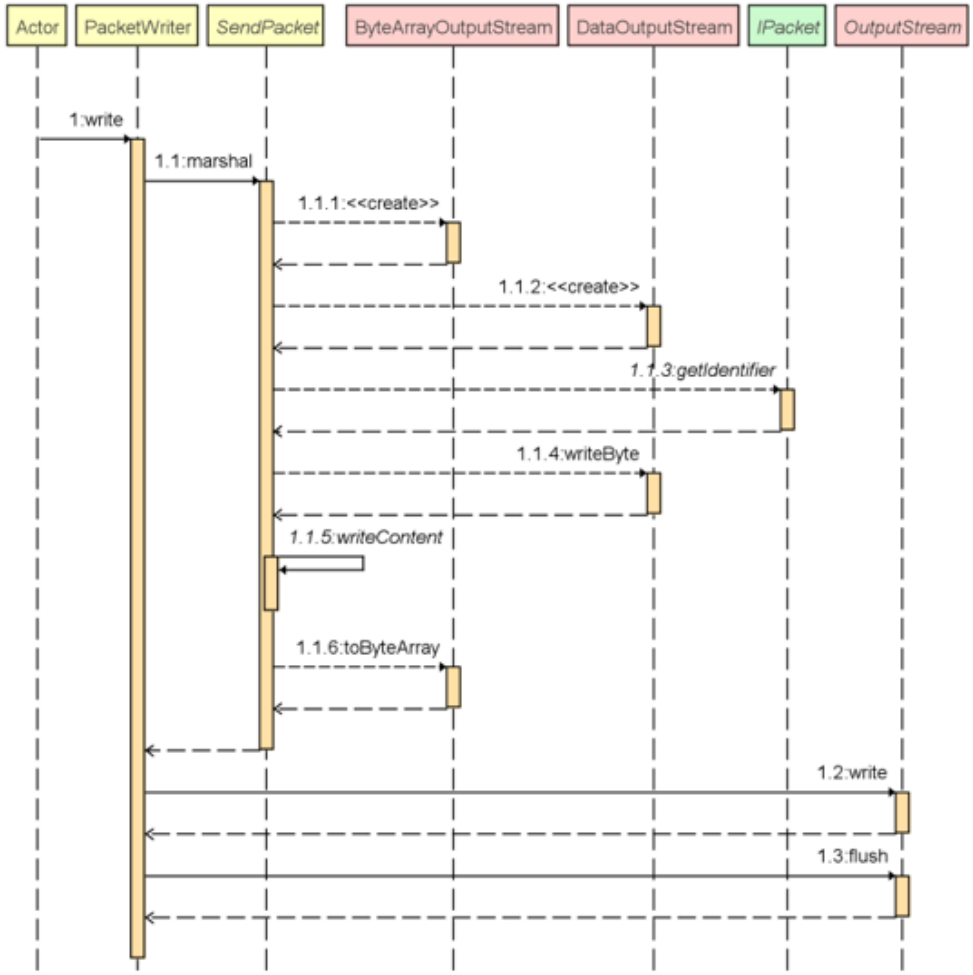

Abbildung 7: Ablauf eines Sendevorgangs

Damit ein Paket gesendet werden kann, muss die dazugehörige Klasse von der Klasse ,,SendPacket" abgeleitet werden und die Methoden ,,getIdentifier" und ,,writeContent" implementieren. ,,getIdentifier" gibt eine für den PaketTypen eindeutige ID zurück. ,,writeContent" schreibt die zum Paket gehörenden Daten (bis auf den Identifier) in einen DataOutputStream und führt damit das Marshalling durch. Die erstellte Instanz des Paketes muss nun an die ,,write"-Methode, des zur entsprechenden Verbindung gehörenden PacketWriters, gegeben werden. Dieser führt die ,,marshal"-Methode des Paketes aus und schreibt die entsprechenden Daten in den OutputStream des TCP-Sockets.

4.2.2 Pakete empfangen

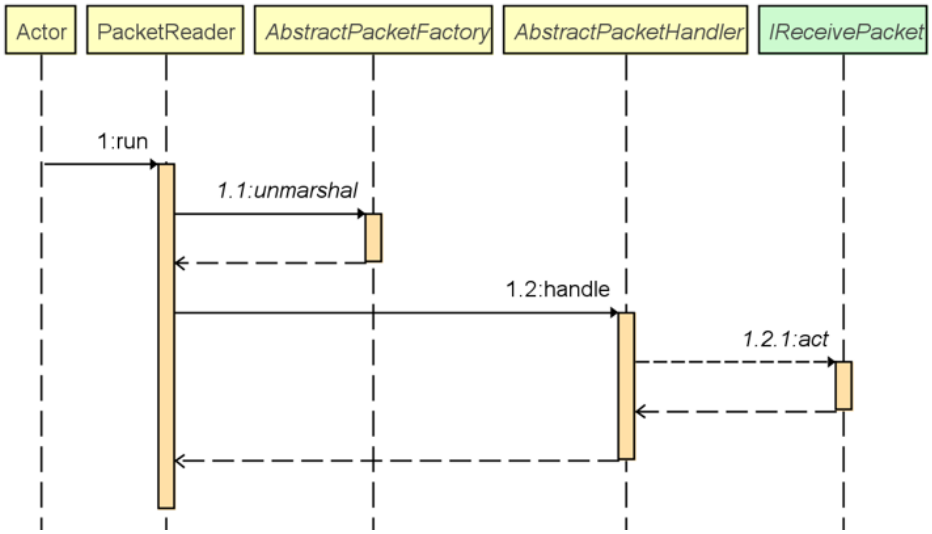

Abbildung 8: Ablauf eines Empfangvorgangs

Das Empfangen von Paketen wird durch den PacketReader bewerkstelligt. Der PacketReader besitzt den DataInputStream eines Sockets und liest aus diesem in einer Endlosschleife Daten aus. Der Ausleseprozess beginnt mit dem Auslesen des Paket-Identifiers. Anschließend biezieht sich der Reader die zum Identifier passende Factory-Klasse, welche das restliche Paket aus den aus dem Stream kommenden Daten zusammenbaut. Da die „read"-Methoden des InputStreams blockierend sind und der PacketReader ein eigener Thread ist, wird aktives Warten vermieden.

Nach dem Zusammenbauen des Paketes wird dieses vom PacketHandler ausgeführt. Der PacketHandler erlaubt, je nach Zustand der Verbindung, nur die Ausführung eines bestimmten Paket-Typen. Manche Paket-Typen benötigen mehr Daten als andere. So gibt es beispielsweise die PreAuth-Pakete, welche vor der Authentifizierung verwendet werden. Hat sich ein Spieler eingeloggt, wird die Verbindung auf eine AuthenticatedConnection hochgestuft und beinhaltet damit auch ein Spieler-Objekt. Sobald dieser Vorgang abgeschlossen wurde, verarbeitet der PacketHandler nur noch Pakete, welche das Interface ILobbyReceivePacket implementieren. So kann verhindert werden, dass beispielsweise ein bereits eingeloggter Nutzer, mit einer AuthenticatedConnection, einen weiteren Account registriert. Tritt ein Spieler einem Spiel bei, wird die Connection auf eine GameConnection hochgestuft und es können nur noch Pakete verwendet werden, welche von IGameReceivePacket abgeleitet sind. Des Weiteren beinhaltet die GameConnection das Spiel-Objekt, auf welches Pakete direkt zugreifen können. Pakete werden von dem PacketReader-Thread ausgeführt und beinhalten eine „act"-Methode, welche das Verhalten des Paketes bestimmt.

4.2.3 Propagierung eines Paketes durch das System

In Abbildung 9 wird an Beispiel eines ShootPackets gezeigt, wie sich Nachrichten durch das System verbreiten. Das ShootPacket-Paket wird von einem Clienten versendet, wenn der Spieler am Zug ist und auf ein gegnerisches Spielfeld klickt. Der Client gibt das ShootPacket-Paket daraufhin an seine Verbindung zum Server, welche das Paket marshalled (beschrieben in Abschnitt 4.2.1

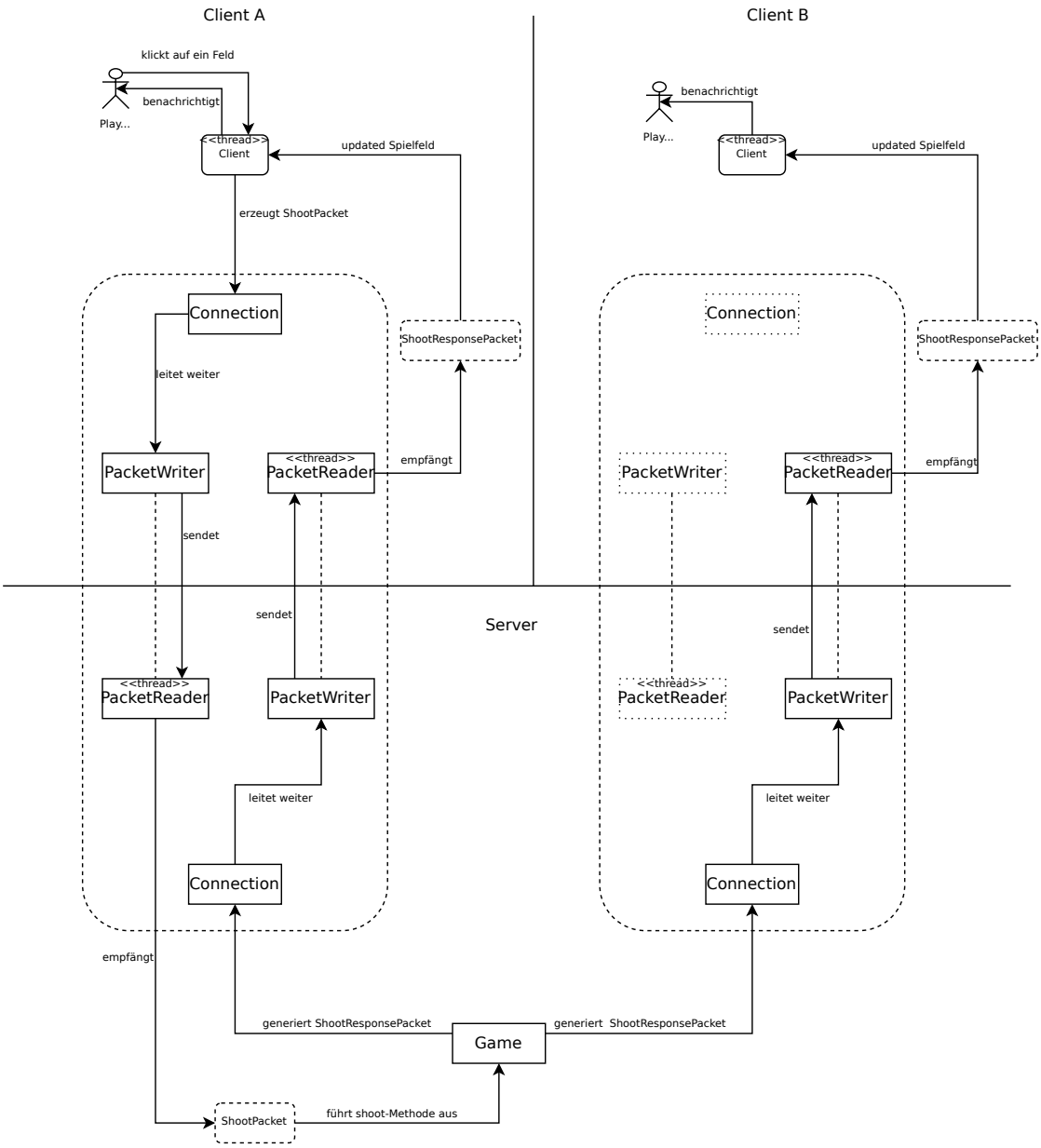

Abbildung 9: Ablauf einer Schuss-Nachricht im System mit zwei Clienten und einem Server

„Pakete senden") und zum Server sendet. All dies geschieht im Client-Thread.

Der Server liest die Daten über den PacketReader, welcher zur Verbindung gehört aus (beschrieben in Abschnitt 4.2.2). Er erstellt ein serverseitiges ShootPacket-Objekt und führt dessen act-Methode aus. Die act-Methode greift auf das Game-Objekt, welches der Verbindung zugeordnet ist zu. Das Game-Objekt verarbeitet daraufhin die Informationen aus dem ShootPacket und reagiert entsprechend der Spiellogik. Danach versendet es an beide Clienten, welche dem Spiel zugeordnet sind, sogenannte ShootResponsePacket-Pakete. Das Verarbeiten solcher Pakete, inklusive dem Behandeln der Spiellogik werden immer im Thread des empfangenden PacketReaders ausgeführt.

Die Clienten lesen jeweils das ShootResponsePacket-Paket vom Server aus und reagieren entsprechend der Daten im Paket. So zeigt der original sendende Client dem Nutzer an, dass dieser ein gegnerisches Feld getroffen oder verfehlt hat, und der andere Client zeigt seinem Nutzer an, dass auf ein Feld von ihm geschossen wurde. Clientseitig werden die Pakete auch von dem zur Verbindung gehörenden PacketReader ausgelesen, allerdings werden die „sichtbaren" Aktionen nicht vom PacketReader-Thread ausgeführt, sondern vom Client-Thread. Dies geschieht, indem aus dem PacketReader-Thread Runnable-Objekte an den Client-Thread übergeben werden.

4.2.4 Heartbeats

Um sicher stellen zu können, dass die Verbindung zwischen Client und Server noch offen ist und nicht beispielsweise durch einen Stromausfall oder einem anderen Störfaktor unsauber unterbrochen wurde, senden sich beide Parteien in regelmäßigen Abständen (5 Sekunden) sogenannte Heartbeat-Pakete zu. Empfängt eine Partei über einen Zeitraum von 10 Sekunden kein Heartbeat-Paket, geht diese davon aus, dass die Verbindung geschlossen wurde und reagiert entsprechend.

4.2.5 Fehlerbehandlung

Mögliche Fehler sind: Eingeben einer falschen Serveradresse, Fehler bei der Authentifizierung, so wie das Abbrechen der Netzwerkverbindung, bzw. das vorzeitige Verlassen des Spieles durch einen Nutzer, so wie das Erhalten eines unerwarteten Pakets. Wird eine ungültige Serveradresse angegeben, so bekommt der Nutzer diesbezüglich eine Meldung (Verbindung fehlgeschlagen). Stimmt die Server Adresse, jedoch Nutzername und/oder Passwort nicht, so wird auf einen fehlgeschlagenen Login hingewiesen. Versucht man einen bereits vorhandenen Nutzer zu registrieren, wird man ebenfalls daraufhin gewiesen. In allen Fällen wird man auf das Login Fenster zurückgeleitet.

Ein Verbindungsabbruch kann sowohl vom Client, als auch vom Server mithilfe des Heartbeats festgestellt werden. In diesen Fällen wird die Verbindung von einer Seite geschlossen, wodurch die andere Seite, spätestens durch den fehlenden Heartbeat, erkennen kann, dass die Verbindung geschlossen wurde.

Verlässt ein Spieler vorzeitig ein Spiel, so wird dies mit einer hervorgehobenen Nachricht im Chat Fenster dargestellt und das Spiel zählt für den verbleibenden Spieler als gewonnen (mit der Anmerkung, dass der Gegner aufgegeben hat).

Eine Verbindung hat verschiedene Zustände, in der nur bestimmte Pakete erhalten werden können. Wird ein Paket außerhalb des entsprechenden Zustandes erhalten, so wird es verworfen.

4.3 Benutzermanagement

Jeder Spieler benötigt einen Account. Dieser kann über die Verbindung im PreAuth-Zustand erstellt werden. Nutzerkonten werden in einer SQLite-Datenbank mit verschlüsseltem Passwort gespeichert. Beim Registrieren wird sichergestellt, dass ein Nutzername nicht mehrfach vergeben wird. Des Weiteren wird für jeden Nutzer eine ID generiert. Hat sich ein Nutzer registriert, wird dieser mit dem Registrierungs-Paket automatisch eingeloggt. Will sich ein Spieler einloggen, muss dieser ein Login-Paket senden. Der Server gleicht die Logindaten mit der Datenbank ab und überprüft sie auf Korrektheit. Anschließend antwortet der Server dem Client und teilt diesem mit, ob der Login-Versuch erfolgreich war. Im Erfolgsfall wird die Verbindung auf eine AuthenticatedConnection hochgestuft und damit dem Benutzer zugeordnet. Falls der Login-Versuch fehlschlägt, wird dem Spieler eine entsprechende Fehlermeldung angezeigt. Um zu verhindern, dass Spieler gegen sich selbst spielen, erlaubt das Benutzermanagement für jeden Benutzer nur eine gleichzeitige Verbindung. Ist ein Benutzer bereits über eine andere Verbindung eingeloggt, schlägt der Login-Versuch fehl.

4.4 Projektaufbau

Das Projekt besteht aus drei Teilprojekten. Zusätzlich zum Client und Server gibt es auch ein Commons-Projekt. Das Commons-Projekt stellt allgemeine Klassen, wie beispielsweise den PacketReader und PacketWriter zur Verfügung. Alle Klassen in diesem Teilprojekt sind in den anderen Projekten verfügbar und werden beim Bauen des Clienten und des Servers automatisch bereitgestellt. Auf diese Weise wird doppelter Code innerhalb der Client- und Server-Projekte vermieden.