MNM
TEAM
Munich Network Management Team

Ludwig-Maximilians-Universität München

Prof. Dr. D. Kranzlmüller
Dr. Karl Fürlinger
Jan Schmidt
Amir Kabouteh

Willkommen in der Gruppenphase des Systempraktikums. Ihre Aufgabe in der Projektphase ist es, einen Client für das Spiel *Reversi* in der Programmiersprache `C` zu entwickeln. Die Übungsblätter werden Sie schrittweise zu diesem Ziel führen.

Der Lehrstuhl stellt im Rahmen des Systempraktikums einen Gameserver und ein Webinterface bereit. Der Gameserver implementiert den Spielablauf, Ihr Client die Spielelogik. Der Gameserver ist gewissermaßen der Spielleiter und somit insbesondere für die Regeleinhaltung verantwortlich, wohingegen Ihr Client in die Rolle eines Mitspielers schlüpft. Über das Webinterface unter `http://sysprak.priv.lab.nm.ifi.lmu.de` können Partien verwaltet werden. Der Gameserver und das Webinterface sind nur aus dem MWN[1] erreichbar (Hinweis: `ssh -L` auf die CIP-Rechner[2]).

Um das Testen Ihres Clients zu erleichtern, bietet das Webinterface zudem die Möglichkeit als menschlicher Mitspieler an einer Partie teilzunehmen, entweder in der Rolle eines Beobachters oder eines Mitspielers. Sie können jetzt testweise eine neue Partie erstellen und zum Einstieg gegen ihre Kommilitonen oder sich selbst spielen. Sie werden einen Einblick gewinnen, wie das Spiel abläuft.

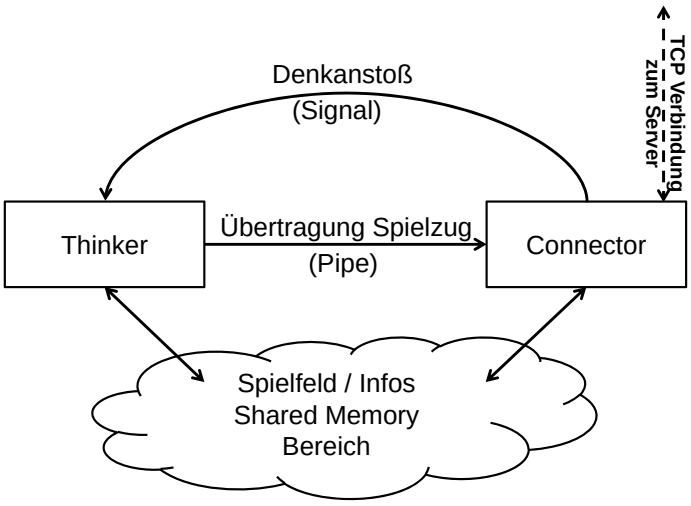

Abbildung 1: Übersicht über den Client

Der von Ihnen zu entwickelnde Client wird im Verlaufe des Systempraktikums schrittweise entwickelt. Maßgeblich besteht er aus zwei Prozessen, dem *Thinker* und dem *Connector*. Der *Connector* übernimmt die Kommunikation mit dem Gameserver und der *Thinker* berechnet den nächsten Spielzug.

Wird der Connector vom Gameserver zum nächsten Spielzug aufgefordert, so legt er alle Informationen, die er vom Gameserver bekommen hat um den nächsten Spielzug zu berechnen, in einen *geteilten Speicherbereich*. Danach sendet der Connector ein *Signal* an den Thinker. Der dadurch „aufgeweckte“ Thinker beginnt nun (mit Hilfe der aus dem *geteilten Speicherbereich* gelesenen Informationen) den nächsten Spielzug zu berechnen. Das Ergebnis der Berechnungen (der Spielzug) sendet der Thinker über eine *unnamed Pipe* an den Connector zurück, welcher den Spielzug an den Gameserver sendet. Abbildung 1 gibt Ihnen einen groben Überblick über den Aufbau des Clients.

[1] Informationen zum MWN: `http://www.lrz.de/services/netz/`
[2] Informationen zu SSH: `http://www.rz.informatik.uni-muenchen.de/FAQ/Aussenzugriff.faq.html/`

1 Reversi

Informationen zum Spiel und die im Praktikum verwendeten Spielregeln sind im Webinterface des Gameservers unter `http://sysprak.priv.lab.nm.ifi.lmu.de/help/reversi/` zu finden.

2 Protokolldefinition des Gameservers

Ihr zu implementierender Client kann mit dem Gameserver über das hier beschriebenen zeilenorientierte Protokoll kommunizieren. Zeilen werden, wie unter Unix üblich, mit einem „newline character“ (`'\n'`) abgeschlossen. Der MNM-Gameserver ist wie folgt zu erreichen:

- *Hostname:* `sysprak.priv.lab.nm.ifi.lmu.de`
- *Port:* `1357` (TCP)

Wenn der Gameserver eine Zeile mit einem `+` als erstes Zeichen schickt, ist dies eine positive Antwort. Im Folgenden ist nur der positive Verlauf einer Kommunikation angegeben. An jedem Schritt kann eine Negativantwort auftreten, diese ist erkennbar an dem `-` als erstes Zeichen der Zeile. Ein `-` ist stets gefolgt von einer Fehlermeldung. Im Anschluss an die Fehlermeldung wird die Verbindung getrennt.

Wenn nicht innerhalb von im Gameserver festgelegten Zeitgrenzen auf Befehle geantwortet oder eine zu lange „Denkzeit“ benötigt wird (s. u.), schickt der Gameserver:
S: - TIMEOUT ⟨⟨ *Begründung* ⟩⟩

Die folgende Protokolldefinition kürzt eine Zeile, welche vom Client an den Gameserver geschickt wird, mit **C:** für *Client* ab. Eine Zeile, welche vom Gameserver an den Client übermittelt wird, wird mit **S:** für *Server* abgekürzt.

In ⟨⟨ ⟩⟩ eingeschlossene Werte werden obligatorisch durch die ihnen entsprechenden Werte ersetzt. Werte, die in [[ ]] eingeschlossen sind, geben optionale Werte an, d. h. sie können auch weggelassen werden.

Das Protokoll gliedert sich in folgenden drei Phasen:

1. *Prolog* – hier wird dem Spiel beigetreten und Informationen über das Spiel ausgetauscht.
2. *Spielverlauf* – hier wird gewartet bis man an der Reihe ist, bzw. das Spiel beendet wird.
3. *Spielzug* – hier übermittelt der Gameserver ein Spielfeld und erwartet einen Spielzug.

Diese Phasen werden in den folgenden Unterkapiteln ausführlich beschrieben. Das Zustandsdiagramm in Abbildung 2 gibt eine grobe Übersicht über den Ablauf der drei Protokollphasen.

2.1 Protokollphase *Prolog*

⟨⟨ *Aufbau der TCP-Verbindung durch Client* ⟩⟩
S: `+ MNM Gameserver` ⟨⟨ *Gameserver Version* ⟩⟩ `accepting connections`
C: `VERSION` ⟨⟨ *Client Version* ⟩⟩
S: `+ Client version accepted - please send Game-ID to join`
C: `ID` ⟨⟨ *Game-ID* ⟩⟩
S: `+ PLAYING` ⟨⟨ *Gamekind-Name* ⟩⟩
S: `+` ⟨⟨ *Game-Name* ⟩⟩
C: `PLAYER` [[*Gewünschte Mitspielernummer*]]
S: `+ YOU` ⟨⟨ *Mitspielernummer* ⟩⟩ ⟨⟨ *Mitspielername* ⟩⟩
S: `+ TOTAL` ⟨⟨ *Mitspieleranzahl* ⟩⟩

Nun kommt für jeden der anderen Spieler die Zeile:
S: `+` ⟨⟨ *Mitspielernummer* ⟩⟩ ⟨⟨ *Mitspielername* ⟩⟩ ⟨⟨ *Bereit* ⟩⟩

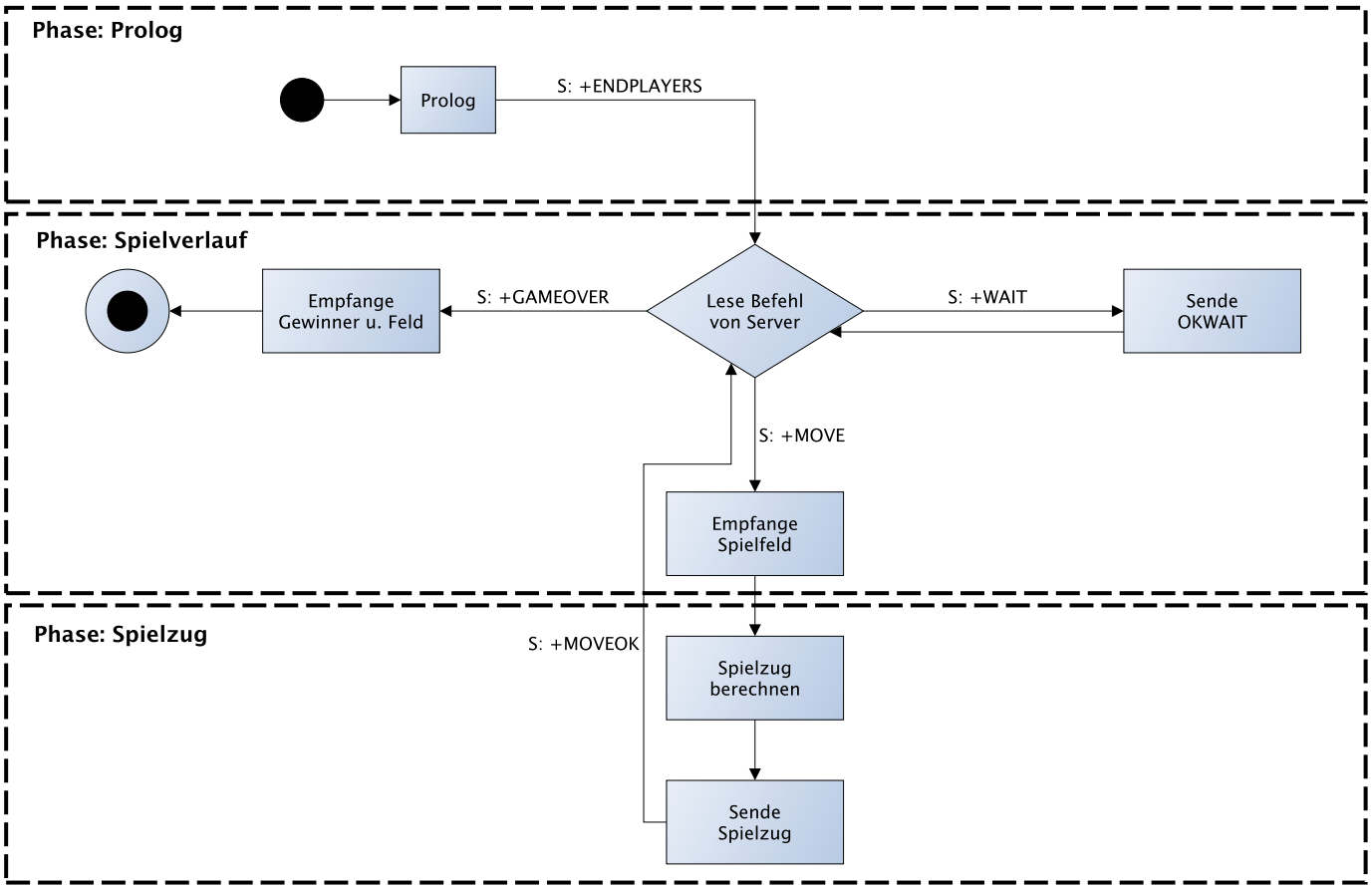

Abbildung 2: Protokollübersicht

S: `+ ENDPLAYERS`

Der Gameserver akzeptiert den Client, wenn die Major-Versionsnummern (links vom Punkt) von ⟨⟨ *Gameserver Version* ⟩⟩ und ⟨⟨ *Client Version* ⟩⟩ gleich sind. Der Gameserver setzt jeweils ein `v` voran. Die Major-Versionsnummer des Gameserver ist dieses Semester immer 2, die Minor-Versionsnummer kann sich jedoch ändern. Kompatibel sind z. B. ein Gameserver mit der Version `v2.1` und ein Client mit der Version `2.42`. Nicht kompatibel wären hingegen `v2.1` und `3.1` oder `v2.1` und `v2.1` (Client hat ein `v` Präfix).

Da der Gameserver nicht nur Reversispielen kann, teilt ⟨⟨ *Gamekind-Name* ⟩⟩ die Spielart mit. Ist diese anders als `Reversi`, muss der Client sich mit einer Meldung, die auf dieses Problem hinweist, beenden.

⟨⟨ *Game-Name* ⟩⟩ und ⟨⟨ *Mitspielername* ⟩⟩ können im Webinterface beim Erstellen eines Spiels angegeben werden, bzw. enthalten Standartwerte, wenn die Angabe ausbleibt.

Wenn man einen Spieler mit einer bestimmten Nummer übernehmen möchte, kann dieser Wunsch mit [[*Gewünschte Mitspielernummer*]] angegeben werden. Lässt man die Mitspielernummer und das Leerzeichen nach `PLAYER` weg, so bekommt man einen freien Computerspieler vom Gameserver zugeteilt.

Die ⟨⟨ *Mitspieleranzahl* ⟩⟩ wird bei einem Reversi-Spiel immer `2` sein.

Ist ein Spieler bereits verbunden, so ist ⟨⟨ *Bereit* ⟩⟩ `1`, ansonsten `0`. Auch wenn noch nicht alle Mitspieler bereit sein sollten, so wird mit der nächsten Protokollphase fortgefahren.

2.2 Protokollphase *Spielverlauf*

In dieser Phase können eine *Idle*, *Move* sowie eine *Game over* Befehlssequenz vorkommen. Wurde das erste Mal in die Spielverlaufsphase gewechselt oder eine der Befehlssequenzen beendet, so entscheidet die nächste Zeile (die im Diagramm an den Kanten annotiert ist) über die nächste abzuarbeitende Befehlssequenz.

2.2.1 *Idle* Befehlssequenz

S: `+ WAIT`
C: `OKWAIT`

`WAIT`-Befehle kommen in regelmäßigen Abständen und müssen rechtzeitig mit einem `OKWAIT` quittiert werden. Ansonsten beendet der Gameserver die Verbindung und die Partie gilt im Turnier als verloren. Wie dem Diagramm in Abbildung 2 zu entnehmen ist, bleiben die `WAIT`-Befehle z. B. in der Protokollphase *Spielzug* oder während der *Move* Befehlssequenzen aus.

2.2.2 *Move* Befehlssequenz

S: `+ MOVE` ⟨⟨ *Maximale Zugzeit* ⟩⟩
S: `+ FIELD` ⟨⟨ *Anzahl Spalten* ⟩⟩`,`⟨⟨ *Anzahl Zeilen* ⟩⟩

Die folgende Zeile wird nun für jede Zeile des Spielfeldes geschickt, beginnend bei der obersten Zeile des Spielfelds:
S: `+` ⟨⟨ Y ⟩⟩ ⟨⟨ $Stein_{1Y}$ ⟩⟩ ⟨⟨ $Stein_{2Y}$ ⟩⟩ ... ⟨⟨ $Stein_{X_{max}Y}$ ⟩⟩

S: `+ ENDFIELD`
C: `THINKING`
S: `+ OKTHINK`

Der `MOVE`-Befehl fordert zum Zug auf. Nach der in Millisekunden angegebenen ⟨⟨ *Maximale Zugzeit* ⟩⟩ muss die anschließende Protokollphase „Spielzug“ abgeschlossen sein. Bitte berücksichtigen Sie bei Ihrer Implementierung die Latenzzeiten der Verbindung. Beachten Sie, dass zwischen ⟨⟨ *Anzahl Spalten* ⟩⟩ und ⟨⟨ *Anzahl Zeilen* ⟩⟩ kein Leerzeichen steht. Ist ein Feld belegt, so ist ⟨⟨ $Stein_{XY}$ ⟩⟩ „`W`“ für den hellen Spieler oder „`W`“ für den dunklen. Ist das Feld frei so ist ⟨⟨ $Stein_{XY}$ ⟩⟩ „`*`“. Zum Beispiel könnte eine übermittelte Zeile `+ 4 * * * W B * * *` lauten.

Der Client muss direkt nach der Übermittlung des Spielfeldes `THINKING` schicken und hat hierfür wenig Zeit. Nach der Gameserver-Antwort `OKTHINK` befinden wir uns in der Protokollphase „Spielzug“.

2.2.3 *Game over* Befehlssequenz

S: `+ GAMEOVER`
S: `+ FIELD` ⟨⟨ *Anzahl Spalten* ⟩⟩`,`⟨⟨ *Anzahl Zeilen* ⟩⟩

Die folgende Zeile wird nun für jede Zeile des Spielfeldes geschickt, beginnend bei der obersten Zeile des Spielfelds:
S: `+` ⟨⟨ Y ⟩⟩ ⟨⟨ $Stein_{1Y}$ ⟩⟩ ⟨⟨ $Stein_{2Y}$ ⟩⟩ ... ⟨⟨ $Stein_{X_{max}Y}$ ⟩⟩

S: `+ ENDFIELD`
S: `+ PLAYEROWON` ⟨⟨ *'Yes' oder 'No'* ⟩⟩
S: `+ PLAYER1WON` ⟨⟨ *'Yes' oder 'No'* ⟩⟩
S: `+ QUIT`
⟨⟨ *Abbau der TCP-Verbindung durch Gameserver* ⟩⟩

Nach einem `GAMEOVER` stellen die weiteren Zeilen den Spielstand dar, mit dem das Partieende erreicht wurde. Dieser wird an alle Mitspieler geschickt. Zudem wird angegeben, welche Mitspieler gewonnen haben. Es ist auch möglich, dass eine Partie in einem Unentschieden endet. In dem Fall ist allen Mitspielern der Gewinnstatus der Partie auf `Yes` gesetzt. Nach `QUIT` beendet der Server die Verbindung.

2.3 Protokollphase *Spielzug*

C: `PLAY` ⟨⟨ *Spielzug* ⟩⟩
S: `+ MOVEOK`

Ein ⟨⟨ *Spielzug* ⟩⟩ ist schlicht die Koordinate eines freien Feldes, auf die der eigene Stein gelegt werden soll (z. B. `E6`). Sollte der übermittelte Zug nicht gültig sein, wird eine entsprechende Fehlermeldung übermittelt und die Verbindung getrennt.

3 Bugreports

Der Gameserver wird jedes Jahr um neue Funktionen und Spiele erweitert. Trotz aller Bemühungen kann es passieren, dass noch Fehler im System sind. Melden Sie diese Bugs bitte auf https://gitlab.lrz.de/sysprak/frontend/issues. Verwenden sie dabei die passende Meldevorlage für ihr Anliegen.

Für organisatorische Anliegen wenden sie sich bitte an sysprak-admin@nm.ifi.lmu.de.