Programmieren
4 Gewinnt Protokoll
Aktualisierte Version

2. Januar 2013

Inhaltsverzeichnis

1 Änderungshistorie

Abschnitt	Änderung
28.12.2012	
3.4. Fehlerfälle	Neuer Abschnitt
4. Nachrichten	Newline als Nachrichtenterminator Aufbau einer Nachricht näher spezifiziert
4.5.2. Spielzug übermitteln	Definition der Feldnummer
4.2.2. Spiellistenübermittlung, 4.5.3. Spielzustandsänderung	Ergänzungen (Werte von Parametern)
02.01.2013	
4.4. Spiel versiegeln	Ergänzungen (Antworten vom Server)
4.5.2. Spielzug übermitteln	Gravitationsrichtung

2 Einführung

Das vorliegende Dokument beschreibt das Protokoll für den Ablauf einer im Netzwerk stattfindenden 4 Gewinnt Session.

Das Design des Protokolls ist dabei so einfach wie möglich gehalten, um eine schnelle und zuverlässige Implementierung zu ermöglichen. Der Datenaustausch findet rein textbasiert statt. Dies soll die Fehlerfindung bzw. Nachvollziehbarkeit der Zustandsänderungen vereinfachen.

Abschnitt 3 beschreibt die Kommunikationsabläufe zwischen den Akteuren, die an Verwalten und Spielen von 4-Gewinnt Sessions beteiligt sind. Abschnitt 4 beschreibt alle im Protokoll verwendeten Nachrichten im Detail.

3 Protokollbeschreibung

Das Protokoll geht von einer Client-Server Architektur aus. Die Kommunikation zwischen den Akteuren erfolgt mittels TCP mit Ausnahme des Hearbeats (siehe Abschnitt 3.3 bzw. Abschnitt 4.5.6), welches über UDP transportiert wird.

Will ein Spieler (der Initiator) ein neues Spiel starten, so kann er es unter einem eindeutigen Namen auf einem Indizierungsserver registrieren. Das Spiel bzw. seine Adresse ist nun für alle Interessen (Clients) über den Index auffindbar.

Neben der Adresse werden alle Metainformationen eines Spiels (IP Adresse, Port, Spielername, Spielstatus, Spielertyp) gespeichert. Der Registrierungsvorgang wird in Abschnitt 3.1 im Detail beschrieben.

Jeder Client darf eine beliebige Anzahl an Spielen eröffnen und ist allein für den Spielzustand und die korrekte Notifzierung des Indizierungsservers verantwortlich. Ein Spiel kann dabei die Zustände *offen* und *laufend* annehmen. Einem *laufenden* Spiel kann nicht mehr beigetreten werden.

3.1 Szenario 1: Spielerstellung und Spielbetritt

Anmerkung: In diesem Szenarion wird davon ausgegangen, dass noch keine Dienste gestartet wurden. Der Initiator möchte ein Spiel hosten. Dazu muss das Spiel auf einem Indizierungsserver registriert werden, damit es andere Spieler finden können. Der Indizierungsserver kann entweder lokal gestartet werden oder für andere Teilnehmer im Netzwerk zur Verfügung stehen. Adressen für Inidizierungsserver müssen a priori bekannt sein[1].

1. Der Initiator startet einen lokalen Indizierungsserver I.
2. Der Initiator registriert das Spiel S mit dem Namen N auf I (siehe Abschnitt 4.1.1).
   (a) I überprüft, ob das Spiel mit dem Namen N bereits registiert wurde. In diesem Fall sendet I eine Fehlermeldung an den Initiator (siehe Abschnitt 4.1.2).
   (b) Ist der Name N frei, bestätigt I die Anfrage (siehe Abschnitt 4.1.2).
3. Client B verbindet sich zu I und sendet eine Anfrage zur Ermittlung aller in I registrierten Spiele (siehe Abschnitt 4.2.1).
4. I sendet eine vollständige Auflistung aller Spiele inklusive deren Metainformationen (siehe Abschnitt 4.2.2).
5. Client B wählt das Spiel S aus und verbindet sich mit dem Initiator.
6. Client B sendet eine Beitrittsanfrage an den Initiator (siehe Abschnitt 4.3.1).
7. Der Initiator sendet entweder eine Beitrittsbestätigung (siehe Abschnitt 4.3.2) oder lehnt die Anfrage ab (beispielsweise wenn das Spiel bereits läuft oder der Beitritt nicht erwünscht ist) (siehe Abschnitt 4.3.2).
8. Im Falle der Annahme wird der Spielzustand auf *laufend* geändert und I über die Zustandsänderung notifiziert (siehe Abschnitt 4.4).

[1] Das Protokoll schreibt nicht vor, wie der Austausch von Adressen zu erfolgen hat. Diese könnten beispielsweise durch Netzwerkscans ermittelt werden.

3.2 Szenario 2: Spielablauf

Basierend auf Szenario 1 wird davon ausgegangen, dass sich zwei Clients in einem Spiel befinden.

1. Der Initiator ermittelt zufällig den Spieler R mit dem ersten Spielzug.
2. Der Initiator sendet an R das Startsignal[2](siehe Abschnitt 4.5.1).
3. Nachdem sich R für einen Zug entschieden hat, wird der Initiator über diesen informiert (siehe Abschnitt 4.5.2).
   (a) Nach einer erfolgreichen Validierung, sendet der Initiator den neuen Spielzustand an alle Spieler (siehe Abschnitt 4.5.3).
   (b) Ist die Validierung nicht erfolgreich, wird das Spiel beendet und alle Spieler inklusive Indizierungsserver über das vorzeitige Spielende informiert (siehe Abschnitt 4.5.5).
4. Der Initiator überprüft ob das Spielende erreicht ist. In diesem Fall werden alle Spieler über den Spielausgang notifiziert (siehe Abschnitt 4.5.4) und das Spiel aus dem Indizierungsserver entfernt (siehe Abschnitt 4.2.3).
5. Andernfalls ist der Gegenspieler G am Zug. Der Ablauf beginnt wieder ab Punkt 3, mit der Bedingung, dass R zu G wird.

3.3 Szenario 3: Ausfälle

Von einem Ausfall können Indizierungsserver und Clients betroffen sein. Folgendes ist zu beachten:

- Solange ein Initiator ein Spiel auf einem Indizierungsserver I registiert hat, muss I innerhalb von 60 Sekunden mindestens ein Heartbeat (siehe Abschnitt 4.5.6) vom Initiator empfangen. Das gleiche gilt mit vertauschten Rollen. Bei Empfang eines Heartbeats wird das Timeout neu gesetzt. Stellt I ein Timeout fest, müssen alle Spiele des Initiators entfernt werden. Um UDP inhärente Paketverluste zu kompensieren, wird empfohlen, alle 10 Sekunden ein Heartbeat zu senden.
- Stellt sich bei einem Spielzustandsupdate an den Indizierungsserver heraus, dass dieser nicht mehr erreichbar ist, werden alle Spieler über das vorzeitige Spielende informiert.
- Stellt der Initiator ein Timeout von 10 Sekunden zum Gegenspieler fest, so wird das Spiel abgebrochen und alle Spieler und der Indizierungsserver über das vorzeitige Spielende informiert.

3.4 Fehlerfälle

Gehen Sie davon aus, dass fehlerhafte bzw. unerwartete Nachrichten empfangen werden können.

Sowohl Client als auch Server dürfen dabei nicht abstürzen, sondern sollen zumindest mit einer aussagekräftigen Fehlermeldung terminieren.

Der Indizierungsserver muss über Fehlertoleranz verfügen, d.h., in der Lage sein während eines Kommunikationsproblems mit einem (fehlerhaften) Client Anfragen von anderen Clients korrekt zu verarbeiten.

[2]Selbstverständlich kann R der Initiator selbst sein.

4 Nachrichten

Alle Nachrichten sind in Plaintext UTF8 codiert. Der Aufbau einer Nachricht basiert auf folgender Konvention:

Header	`nachrichten_name`
Parameter	$Parameter_1, Parameter_2, ..., Parameter_N$
Richtung	$A \rightarrow B$

Das Ende einer Nachricht wird mit *Newline* gekennzeichnet. *Header* bezeichnet den eindeutigen Namen (bzw. Typ) der Nachricht, welcher den Parametern vorangestellt wird. Die Anzahl der Parameter ist abhängig vom Typ der Nachricht. Nicht jede Nachricht beinhaltet Parameter. Falls der Nachrichtentyp *Parameter* beinhaltet, so folgen diese unmittelbar nach einem Semikolon. Zeichenketten, die numerische Werte repräsentieren, dürfen keine Whitespaces enthalten. Parameter werden durch Semikolons (U+003B) getrennt.

Beispiel:
Ein Initiator möchte folgendes Spiel mit einem Spielfeld basierend auf 7 Spalten, 6 Zeilen und 4 Ebenen auf einem Indizierungsserver registrieren. Die vom Initiator an den Indizierungsserver zu übermittelnde Nachricht ist nachfolgend dargestellt:

```
register_game;martin;martin's game;7;6;4;198.78.202.118;80\n
```

4.1 Spielregistrierung

4.1.1 Anfrage zur Registrierung

Header	`register_game`
Parameter	Spielername, Spielname, Breite, Höhe, Tiefe, IP-Adresse, Port
Richtung	Initiator → Indizierungsserver

4.1.2 Antwort auf Registrierungsanfrage

Spiel wurde registriert:

Header	`register_success`
Parameter	Spiele GUID
Richtung	Indizierungsserver → Initiator

Spiel konnte nicht registriert werden:

Header	`register_failed`
Parameter	Erklärung
Richtung	Indizierungsserver → Initiator

4.2 Spiellisten

4.2.1 Spieleliste anfordern

Header	`request_game_list`
Parameter	Keine
Richtung	Client → Indizierungsserver

4.2.2 Spieleliste übermitteln

Header	`answer_game_list`
Parameter	Anzahl der Spiele S, (Status, Spielername, Spielname, Breite, Höhe, Tiefe, IP-Adresse, Port) $\times$ S
Richtung	Indizierungsserver → Client

Status ist entweder der String `Open` für offene Spiele oder `Sealed` für bereits laufende Spiele.

4.2.3 Spiel aus Liste entfernen

Header	`unregister_game`
Parameter	Spiele GUID
Richtung	Initiator → Indizierungsserver

Antwort bei fehlgeschlagener Deregistrierung:

Header	`unregister_game_failed`
Parameter	Erklärung
Richtung	Indizierungsserver → Initiator

Antwort bei erfolgreicher Deregistrierung:

Header	`unregister_game_success`
Parameter	
Richtung	Indizierungsserver → Initiator

4.3 Spielbetritt

4.3.1 Anfrage zu Spielbetritt

Header	`join_game`
Parameter	Eigener Spielername, Spielname, Protokollversion
Richtung	Gegenspieler → Initiator

Protokollversion ist entweder der String `V1` oder `V2`. Die Protokollversion entscheidet darüber, welche Informationen beim Spielzugaustausch übermittelt werden (siehe Abschnitt 4.5.3).

4.3.2 Antwort auf Spielbetrittsanfrage

Spieler wird akzeptiert:

Header	`join_game_success`
Parameter	Tatsächliche Protokollversion
Richtung	Initiator → Gegenspieler

Unterstützt der Server nur Protokollversion `V1`, aber der Client fordert `V2` an, so kann die tatsächliche Protokollversion vom Server auf `V1` zurückgesetzt werden. Somit muss `V1` auf jeden Fall als Fallback unterstützt werden, `V2` ist optional.
Spieler wird nicht akzeptiert:

Header	`join_game_failed`
Parameter	Erklärung
Richtung	Initiator → Gegenspieler

4.4 Spiel versiegeln

An den Indizierungsserver wird folgende Nachricht gesendet:

Header	`seal_game`
Parameter	GUID
Richtung	Initiator → Indizierungsserver

Bei erfolgreicher Verarbeitung antwortet der Server mit der Nachricht `seal_game_success` (ohne Parameter). Im Fehlerfall antwortet der Server mit der Nachricht `seal_game_failed` mit einem Parameter, der Fehlerursache.

4.5 Spielablauf

4.5.1 Startsignal

Header	`start_game`
Parameter	Keine
Richtung	Initiator → Gegenspieler, Initiator → Initiator

4.5.2 Spielzug übermitteln

Header	`move`
Parameter	Feldnummer
Richtung	Gegenspieler → Initiator

Für die Feldnummer eines 4-Gewinnt-Spiel mit der Tiefe T, Höhe H, Breite B gilt folgender Zusammenhang:

$$Spielbrett_{3D}[k,i,j] = Spielbrett_{1D}[k \times B \times H + i \times B + j] = Spielbrett_{1D}[Feldnummer]$$

wobei sich Indizes k, i und j jeweils auf Tiefe, Höhe und Breite beziehen. Es gilt: $0 \leq k < T$, $0 \leq i < H$, $0 \leq j < B$, sowie $0 \leq Feldnummer < T \times B \times H$ und $T \geq 0$[3], $B, H > 0$.
Die Gravitation wirkt entlang der Höhenachse. Koordinaten mit der Höhe h ($h \geq 0$) müssen vor Koordinaten der Höhe $h + 1$ besetzt werden.

[3] T=0 für ein zweidimensionales 4-Gewinnt-Spiel

4.5.3 Antwort auf Spielzugübermittlung

Abhängig von der unterstützten Protokollversion wird als Antwort auf einen gültigen Spielzug entweder der gesamte Spielzustand transferiert (Protokollversion V_1) oder die Veränderung zum letztgültigen Zustand (Protokollversion V_2).
Antwort bei Protokollversion V_1:

Header	`synchronize_game_board`
Parameter	Anzahl der Felder F, (Feldnummer, Feldzustand) $\times$ F
Richtung	Initiator $\rightarrow$ Gegenspieler

Antwort bei Protokollversion V_2:

Header	`updated_game_board`
Parameter	Feldnummer, Feldzustand
Richtung	Initiator $\rightarrow$ Gegenspieler

Der Feldzustand kann folgende Werte annehmen:

- 0 für frei
- 1 für belegt durch Initiator
- 2 für belegt durch Gegenspieler

Ein Spielzug ist genau dann gültig, wenn sich die Feldnummer auf ein Feld innerhalb des Spielbrettes bezieht und ein Stein gesetzt werden kann. Antwort bei ungültiger Platzierung:

Header	`moved_failed`
Parameter	Erklärung
Richtung	Initiator $\rightarrow$ Gegenspieler

4.5.4 Spiel beenden

Header	`end_game`
Parameter	Spielausgang
Richtung	Initiator $\rightarrow$ Gegenspieler

Spiel wurde erfolgreich beendet. Spielausgang kann dabei einen der folgenden Werte annehmen:

- 0 für unentschieden
- 1 für Initiator ist Gewinner
- 2 für Gegenspieler ist Gewinner

4.5.5 Spiel vorzeitig beenden

Header	`abort_game`
Parameter	Erklärung
Richtung	Initiator $\rightarrow$ Gegenspieler oder Gegenspieler $\rightarrow$ Initiator

Spiel musste aufgrund eines Fehlers beendet werden. Erklärung ist ein beliebiger Text, der die Ursache des Fehlers beschreibt.

4.5.6 Heartbeat

Header	`heartbeat`
Parameter	Keine
Richtung	Initiator → Indizierungsserver bzw. Indizierungsserver → Initiator