Netzwerkkommunikation

Protokoll und Verhalten

Raphael Kreft

11. Mai 2020

Inhaltsverzeichnis

1 Allgemeines Kommunikationsschema

Grundlage ist die Kommunikation über das TCP(Transport Control Protocol). Die Clients können sich mit dem Server verbinden, wobei auf diesem für jede Client Verbindung ein einzelner Thread gestartet wird, der diese verwaltet (Die Exekutor-Klasse). Über diese Verbindungen werden die Im Protokoll festgelegten Sequenzen übertragen und dann auf dem Server bzw. Client verarbeitet.
Jeder Client verwaltet nur eine Verbindung zu dem jeweiligen Server.
Die Sequenzen werden so validiert, sodass nur Funktionalität aufgerufen werden kann, zu der Client und Server auch Zugriff benötigen. Z.B ist in Lobbys nur der Zugriff auf die in der Lobby wichtige Funktionalität möglich.

Der Präfix GET Ein Solcher Befehl im Protokoll fragt Änderungen bzw. Informationen beim Kommunikationspartner an. Bei dem der den Befehl erhält. Diese werden dann durch zurücksenden einer oder mehrerer SET Anweisungen auf dem ursprünglich anfragenden System ausgeführt.

Der Präfix SET : Ein solcher Befehl löst eine direkte Änderung durch einen Funktionsaufruf aus. Bei dem der den Befehl erhält. Es wird keine Antwort erwartet.

1.1 Befehlsübertragung

Die Sequenzen des Protokolls werden von den Argumenten mit einem Unterstrich „_“getrennt, ein gesendeter Befehl sieht also folgendermaßen aus:

Befehl_ Argument1_ Argument2_ ..._ Argumentn

1.2 Validierung

Die Klasse *Packager* entpackt und packt die zu sendenden oder gesendeten Sequenzen. Die Anzahl der Argumente ist im Protokoll festgelegt. Dabei werden genausoviele Argumente an den Unterstrichen getrennt wie nötig sind; Die restlichen Unterstriche gehen in das letzte Argument mit ein. Somit ist sichergestellt, dass die Anzahl der Argumente nicht das Maximum des Befehls überschreitet. außerdem prüft der *Packager*, ob die richtige Anzahl erreicht wird, d.h. nicht zu wenig Argumente gesendet wurden.

Vor der Ausführung eines jeden Befehls wird außerdem geprüft, ob der Befehl ausführbar ist bezogen auf den aktuellen Zustand des Servers, der Lobby oder des Spiels. Wenn nicht wird der Befehl nicht ausgeführt.

2 Kommunikation Im Servermenü

Verbindet sich ein Client mit einem Server, befindet er sich zunächst im Servermenü und ist keiner Lobby zugeteilt. Von dort aus kann er Lobbys verwalten und offene, laufende und abgeschlossene Spiele betrachten sowie laufenden Spielen zuschauen. Außerdem kann gechattet werden und dir rangliste betrachtet werden.

2.1 Aufrufbare Funktionalität auf dem Server durch den Client

SET-Anweisungen:

- Nickname setzen
- Text in einen Chat senden
- Eine Lobby Erstellen
- Eine Lobby betreten
- Server verlassen
- Einem Spiel zuschauen

2.2 Aufrufbare Funktionalität auf dem Client durch den Server

SET-Anweisungen:

- Du bist im Servermenü
- Neuer Text in einem Chat
- Ein Neuer Client hat sich verbunden
- Ein Client hat den Server verlassen
- Ein Client hat seinen Nicknamen geändert
- Änderung auf der Rangliste!
- Ein neues abgeschlossenes Spiel auf dem Server!
- Eine Neue Lobby wurde erstellt
- Statusänderung einer Lobby
- Eine Lobby wurde gelöscht
- Dein Text wurde erfolgreich gesendet/oder nicht
- Du wurdest der Lobby xy hinzugefügt oder nicht! (geschafft? als Argument + warum)

2.3 Protokoll Clientseite

MISV(MenueInServerView)
Argumente: Nickname Anfangs vom Server vergeben
Beschreibung: Teilt dem Client mit, dass dieser sich erfolgreich verbunden hat und sich im Servermenü befindet.
Funktionalität: Setzt die Scene und den Anfangsnicknamen
Beispiel: MISV_ initialerName

TXTR(TextRecieve)
Argumente: privat/server/lobby, von wem, Nachricht
Beschreibung: Teilt dem Client mit, dass ein neuer Text gesendet wurde, wenn dieser unter den Empängern ist und im Richtigen Menü bzw Spiel um die Nachricht sehen zu können.
Funktionalität: Updatet das Chatfenster
Beispiel: TXTR_ @tim_ oli_ hallo, TXTR_ server_ oli_ hallo, TXTR_ lobby_ oli_ hallo

NCON(NewClientOnServer)
Argumente: nickname
Beschreibung: Teilt dem Client mit, dass sich ein neuer Client mit dem Server verbunden hat
Funktionalität: Aktualisieren der Liste von Clients und der benötigten Grafikelemente
Beispiel: NCON_ Matias

CLSV(ClientLeftServer)
Argumente: nickname
Beschreibung: Teilt dem Client mit, dass ein anderer Client den Server verlassen hat
Funktionalität: Aktualisieren der Liste von Clients und der benötigten Grafikelemente
Beispiel: CLSV_ Matias

CCNN(ClientChangedNickName)
Argumente: oldnickname, newnickname
Beschreibung: Teilt dem Client mit, dass ein anderer Client seinen Nicknamen geändert hat
Funktionalität: Aktualisieren der Clientinformation und der benötigten Grafikelemente
Beispiel: CCNN_ tim_ derBeste

NHSE(NewHighscoreEntry)
Argumente: nickname
Beschreibung: Teilt dem Client mit, dass es einen neuen Eintrag in der Highscoreliste gibt. Dies passiert wenn ein Spieler zum ersten mal ein Spiel gewinnt
Funktionalität: Aktualisieren der Highscore-Liste und der benötigten Grafikelemente bei Bedarf
Beispiel: NHSE_ tim

CHSE(ChangeHighScorelistEntry)
Argumente: nickname
Beschreibung: Teilt dem Client mit, dass es eine Änderung in den Spielerleiszungen/Highscoreliste gibt
Funktionalität: Aktualisieren des entsprechenden Highscorelisteneintrags(Siegeszahl + 1) und der benötigten Grafikelemente bei Bedarf
Beispiel: CHSE_ tim

NFGA(NewFinishedGame)
Argumente: Lobbyname, gewinnernickname
Beschreibung: Teilt dem Client mit, dass ein neues abgeschlossenes Spiel auf dem Server gibt
Funktionalität: Aktualisieren der Liste von abgeschlossenen Spielen und der benötigten Grafikelemente
Beispiel: NFGA_ lobby1_ oli

NLOS(NewLobbyOnServer)
Argumente: Lobbyname, anzahlspieler
Beschreibung: Teilt dem Client mit, dass es eine neue Lobby gibt
Funktionalität: Aktualisieren der Liste von Lobbys und der benötigten Grafikelemente
Beispiel: NLOS_ lobby2_ 2

ALDS(ALobbyDeletedonServer)
Argumente: Lobbyname
Beschreibung: Teilt dem Client mit, dass eine Lobby gelöscht wurde
Funktionalität: Aktualisieren der Liste von Lobbys und der benötigten Grafikelemente
Beispiel: ALDS_ lobby2

ALCS(ALobbyChangedStatus)
Argumente: lobbyname, ingame?, anzahlspieler
Beschreibung: Teilt dem Client mit, dass eine Lobby Ihre Eigenschaften verändert. Also die Anzahl der Mitglieder oder, ob Sie sich im Spiel befindet, jeweils eins der Argumente leer lassen du nur eine Änderung auf Einmal gesendet wird
Funktionalität: Aktualisieren der Information über Lobby und der benötigten Grafikelemente
Beispiel: ALCS_ true_ , ALCS_ _ 2
****Status******
TCMS(TextChatMessageStatus)
Argumente: status(success or fail)
Beschreibung: Teilt dem anfragenden Client mit, ob das Senden eines Textes geklappt hat
Funktionalität: Aktualisieren der ChatListen
Beispiel: TCMS_ fail

FTCL(FailedToCrateLobby)
Argumente: errormsg
Beschreibung: Teilt dem Client mit, dass das erstellen der Lobby niht geklappt hat und warum. Im Efolgsfall erfolgt Aktualisierung über NLOS
Funktionalität: Nutzer benachrichtigen und Aktualisieren der benötigten Grafikelemente
Beispiel: CWCN_ neuerName

FTJL(FailedToJoinLobby)
Argumente:
Beschreibung: Teilt dem Client mit, dass das Betreten einer Lobby nicht geklappt hat. Im Efolgsfall erfolgt Beitrittsmeldung YAIL
Funktionalität: Nutzer benachrichtigen und Aktualisieren der benötigten Grafikelemente
Beispiel: FTJL

YASP(YouAreSpectating)
Argumente: -
Beschreibung: Teilt dem Client mit, dass der sich nun im Spectatormodus befindet.
Funktionalität: Die Szene wird gesetzt und die Bedienelemente deaktiviert.
Beispiel: YASP

2.4 Protokoll Serverseite

CWJL(ClientWantstoJoinLobby)
Argumente: Lobbyname
Beschreibung: Der Client schickt hiermit eine Anfrage zum Betreten einer Lobby an den Server
Funktionalität: Prüfen auf Machbarkeit, wenn ja: Daten aktualisieren und Clients informieren. Wenn nicht: Fehlermeldung an anfragenden Client senden
Beispiel: CWJL_ meineLobby

CWCL(ClientWantstoCreateLobby)
Argumente: Lobbyname
Beschreibung: Client schickt Anfrage zum erstellen einer neuen Lobby
Funktionalität: Prüfen auf Machbarkeit, wenn ja, Daten aktualisieren, Clients informieren. Wenn nicht: Fehlermeldung an anfragenden Client
Beispiel: CWCL_ meineLobby

CWLS(ClientWantstoLeaveServer)
Argumente:
Beschreibung: Client teilt mit, dass er den Server verlässt
Funktionalität: Daten und Objekte anpassen/löschen und alle clients Informieren, socket schliessen (clientsocket wartet bis server den Socket schliesst)
Beispiel: CWLS

CWST(ClientWantstoSendText)
Argumente: privat/server/lobby, message
Beschreibung: Client schickt Anfrage zum versenden eines Textes
Funktionalität: Prüfen auf Machbarkeit, wenn ja, Daten aktualisieren, Clients informieren. Wenn nicht: Fehlermeldung an anfragenden Client
Beispiel: CWST_ user@tim_ hallo, CWST_ server_ hallo, CWST_ lobby_ hallo

CWCN(ClientWantstoChangeNickname)
Argumente: newnickname
Beschreibung: Client schickt Anfrage zum Ändern seines Nicknames
Funktionalität: Prüfen ob einzigartig: wenn der Name nicht eindeutig ist wird ein ähnlicher, einzigartiger gefunden und gesetzt. Alle Clients werden über die Änderung informiert
Beispiel: CWCN_ neuerName

CWCN(ClientWantstoSpectate)
Argumente: Lobbyname
Beschreibung: Client schickt Anfrage zum Beobachten eines laufenden Spiels
Funktionalität: Prüfen ob Gültig(Lobby muss im Spiel sein). und ausführen der entprechenden Logik, die den Client das Spiel beobachten lässt
Beispiel: CWTS_ testlobby

3 Kommunikation in Lobby

Betritt ein Client die Lobby oder kehrt zu ihr zurück aus einem Spiel, so wird der Server SET- Anweisungen senden um den Status der Lobby, also die vorhandenen Mitglieder, den Status zur Spielbereitschaft aller Clients und den bisherigen Chat beim Client gesamtzuaktualisieren. Danach kann der Client Anfragen an den Server schicken und wird stetig über Änderungen des Lobbyzustandes und seiner Variabeln aktualisiert. Die Daten die Im Servermenü wichtig waren interessieren nicht mehr und werden vom Server auch nicht mehr auf den Clients aktualisiert.

3.1 Aufrufbare Funktionalität auf dem Server durch den Client

SET-Anweisungen:

- Text in einen Chat Senden(Argument an wen)
- Lobby verlassen
- Toggle Ready

3.2 Aufrufbare Funktionalität auf dem Client durch den Server

SET-Anweisungen:

- Neuer Text in Chat
- Ein Client hat die Lobby verlassen
- Ein Client hat die Lobby betreten
- Ein Client hat seinen Bereitschaftsstatus geändert
- Das Spiel Startet
- Lobby erfolgreich verlassen
- Du bist in der Lobby!

3.3 Protokoll Clientseite

ACLL(AClientLefttheLobby)
Argumente: nickname
Beschreibung: Der Server teilt mit, dass ein Client die Lobby verlassen hat inder man sich befindet
Funktionalität: Client aktualisiert die Lobbydaten und Grafikelemente des GUI
Beispiel: ACLL_ tim

ACJL(AClientJoinedtheLobby)
Argumente: nickname
Beschreibung: Der Server teilt mit, dass ein Client die Lobby betreten hat inder man sich befindet
Funktionalität: Client aktualisiert die Lobbydaten und Grafikelemente des GUI
Beispiel: ACJL_ Raphael

ACCS(AClientChangeditsStatus)
Argumente: nickname
Beschreibung: Der Server teilt mit, dass ein Client seinen Spielfertigkeitsstatus geändert hat inder man sich befindet
Funktionalität: Client aktualisiert die Lobbydaten des betreffendenClients und Grafikelemente des GUI
Beispiel: ACSS_ Oli

YAIL(YouAreInaLobby)
Argumente: lobbyname
Beschreibung: Der Server teilt mit, man sich in der Lobby befindet und sich auf die Gesamtaktualisierung einstellen kann
Funktionalität: Client bereitet sich auf Empfang der Daten vor (GUI und Datenobjekte)

Beispiel: YAIL_ lobby1

TGST(TheGameStarts)
Argumente:
Beschreibung: Der Server teilt mit, dass das Spiel der Lobby beginnt
Funktionalität: Client aktualisiert teilt es dem user mit setzt die Scene
Beispiel: TGST
Das Empfangen von Texten wird mit schon deklarierten Befehlen geregelt! ****Status****
YLTL(YouLeftTheLobby)
Argumente:
Beschreibung: Der Server teilt mit, dass du erfolgreich die Lobby verlassen hast
Funktionalität: Client wartet auf Benachrichtigung im Servermenü zu sein
Beispiel: YLTL

Der Textchat Status wird durch schon beschriebene Befehle geregelt!

3.4 Protokoll Serverseite

CWTR(ClientWantstotoggleReady)
Argumente:
Beschreibung: Der Client teilt mit sienen Spielbereitschaftsstatus ändern zu wollen.
Funktionalität: Server ändert Status und informiert alle Clients, aktualisiereung erfolgt auch für anfragenden Client
Beispiel: CWTR

CWLL(ClientWantstoLeaveLobby)
Argumente:
Beschreibung: Der Client stellt die Anfrage die aktuelle Lobby zu verlassen, auch wenn er inGame ist
Funktionalität: Server prüft Machbarkeit, ändert Daten und informiert alle Clients in der Lobby, sowie den anfragenden mit YLTL, Reaktion je nachdem ob Lobby inGame oder nicht.
Beispiel: CWLL

4 Kommunikation Im Spiel

Während dem Spiel verwaltet der Server den Spielverlauf, er teilt den Clients mit Wer am Zug ist, prüft die Gültigkeit der Spielzüge und Verteilt jede Spielbewegung an alle an dem Spiel teilnehmenden Clients. Beim betreten des Spiels und bei Verbindungsunterbruch wird die Spielinformation auf jedem Client gesamtaktualisiert. Ist man als Zuschauer in einem Spiel, so kriegt man alle Informationen zum Spiel, kann jedoch keine Operationen ausführen, bis auf den Zuschauermodus zu verlassen.

4.1 Aufrufbare Funktionalität auf dem Server durch den Client

GET-Anweisungen:

- Ich möchte die Runde beenden
- Mögliche Felder zum Bewegen?
- ich möchte diesen Spielzug ausführen
- Gamechat Text senden
- Ich möchte das Spiel verlassen!
- Ich möchte schummeln!

4.2 Aufrufbare Funktionalität auf dem Client durch den Server

SET-Anweisungen:

- Neuer Text im Chat
- Spielzug ist gültig/ungültig
- Ein Neues Objekt auf der Spielkarte
- Objekt von Spielkarte entfernt
- Änderung des Münzkontos
- Änderung des Einkommens pro Runde
- Spieler XY Am Zug
- Restzeit des Zuges
- Ein Feld auf der Karte wechselt den Besitzer
- Spieler XY hat spiel verlassen
- Das Spiel ist beendet

- Neue Spielkarte
- Spiel erfolgreich verlassen
- Neuer Spieler ist am Zug
- Diese Felder sind für die Bewegung möglich
- Spieler hat Spiel verlassen!
- Du bist im Spiel!

4.3 Protokoll Clientseite

Auch Hier wird die Textchatfunktionalität über schon deklarierte Befehle geregelt!!

NEMA(NewMap)
Argumente: sequenz von 0/1 ob begehbar
Beschreibung: Der Server schickt die neue Spielkarte(an alle Clients)
Funktionalität: Client aktualisiert die Spieldaten und Grafikelemente des GUI
Beispiel: NEMA_ 01111100011011100....

FCOW(FieldChangesOWner)
Argumente: FeldID, an wen
Beschreibung: Der Server teilt mit, dass ein Spielfeld den Besitzer gewechselt hat
Funktionalität: Client aktualisiert die Spielfelddaten und Grafikelemente des GUI
Beispiel: FCOW_ 101_ tim

NOOM(NewObjectOnMap)
Argumente: FeldID, was(Gebäude, Einheit), level
Beschreibung: Der Server teilt mit, dass es ein neues Objekt auf dem Spielfeld gibt
Funktionalität: Client aktualisiert die Spielfelddaten und Grafikelemente des GUI
Beispiel: NOOM_ 100_ knight_ 1, NOOM_ 101_ tower_ (Tower haben noch keine Level)

OOMR(ObjectOnMapRemoved)
Argumente: FeldID
Beschreibung: Der Server teilt mit, dass es eine Veränderung bei einem Objekt auf dem Spielfeld gibt
Funktionalität: Client aktualisiert die Spielfelddaten und Grafikelemente des GUI
Beispiel: OOMR_ 200

YCBC(YourCoinBalanceChanged)
Argumente: newBalance
Beschreibung: Der Server teilt mit, sich dein Münzkontostand geändert hat. Dies Geschieht zu Rundenbeginn und beim Kauf von Einheiten

Funktionalität: Client aktualisiert die Münzkontodaten und Grafikelemente des GUI
Beispiel: YCBC_ 15 (Du hast nun 15 Münzen)

BCNR(BalanceChangeNextRound)
Argumente: Balance-Change
Beschreibung: Der Server teilt mit, wie sich das Konto Anfang der nächsten Runde zum aktuellen Zeitpunkt ändern würde.
Funktionalität: Client aktualisiert die Daten und Grafikelemente des GUI
Beispiel: BCNR_ -20

NPOT(NewPlayerOnTurn)
Argumente: nickname(wer)
Beschreibung: Der Server teilt mit, dass ein bestimmter Spieler am Zug ist oder du selber.
Funktionalität: Client teilt dies dem nutzer mit.
Beispiel: NPOT_ tim

GEND(GameEND)
Argumente: wer hat gewonnen(nickname)
Beschreibung: Der Server teilt mit, das Spiel vorbei ist und wer der Gewinner ist
Funktionalität: Client aktualisiert Teilt dies dem Nutzer mit und Wartet auf Rückkehr in Lobbybefehl
Beispiel: GEND_ Matias

LTOT(LeftTimeOfTurn)
Argumente: Zeit
Beschreibung: Der Server teilt die restliche Zeit eines Spielzugs eines Spielers mit.
Funktionalität: Client synchrinisiert mit eigenem Timer!
Beispiel: LTOT_ 2016

PFOR(PossibleFieldsOfRequest)
Argumente: FeldIDs als liste mit , getrennt
Beschreibung: Der Server teilt dem Client die Möglichen Felder einer Bau- oder Bewegungs-Operation mit
Funktionalität: Client umrandet die Begehbaren Felder
Beispiel: PFOR_ 001,102,103,104

4.4 Protokoll Serverseite

CWFT(ClientWantstoFinishTurn)
Argumente:
Beschreibung: Der Client fragt an, ob er die Runde beenden kann
Funktionalität: Server prüft Machbarkeit, aktualisiert Speildaten und Informiert alle

Clients über weiteren Spielverlauf. Wenn nicht: Statusmeldung SCET an Client der anfragt
Beispiel: CWFT

CWRB(ClientWantstoRequestBuy)
Argumente: was? -> Kampfeinheit oder Gebäüde
Beschreibung: Der Client Fragt beim Server an auf welchen Feldern er eine Einheit oder ein Gebäude plazieren könnte.
Funktionalität: Server prüft welche Felder für diesen Spieler zum Setzten einer Einheit/Gebäude in Frage kommen und gibt diese an den Client zurück.
Beispiel: CWRB_ knight, CWRB_ tower

CWRM(ClientWantstoRequestMove)
Argumente: welche Einehit? -> FeldId wo Einheit ist
Beschreibung: Der Client Fragt beim Server an auf welche felder sich die Einheit Bewegen kann.
Funktionalität: Server prüft welche Felder für diesen Spieler zum Setzten bewegen der fraglichen Einheit in Frage kommen und gibt diese an den Client zurück.
Beispiel: CWRM_ 101

CWPI(ClientWantstoPlaceItem)
Argumente: was? -> Kampfeinheit oder Gebäüde, wo? -> FeldId
Beschreibung: Der Client Fragt das Bauen eines Gebäudes/einer Einheit oder das Upate einer solchen an.
Funktionalität: Veranlasst bei Gültigkeit der Operation das Setzten/Aufrüsten. Das Aufrüsten passiert, wenn das Feld was angefragt ist, von einer Einheit besetzt ist und das argument knight gegeben wurde
Beispiel: CWPI_ knight_ 100, CWPI_ tower_ 101

CWTM(ClientWantstoMove)
Argumente: von wo, nach wo (FeldIds)
Beschreibung: Der Client Fragt eine Bewegung einer Einheit an.
Funktionalität: Server Prüft ob die Bewegung Gültig ist, und die Einheit dem Spieler gehört und führt Sie dann aus. Alle Clients werden informiert.
Beispiel: CWTM_ 001_ 004

CWTC(ClientWantstoCheat)
Argumente: Welcher Cheat (0 oder 1)
Beschreibung: Der Client Fragt an zu schummeln.
Funktionalität: Server prüft die Gültigkeit(nur möglich während eigenem Zug).
Cheat 1: Erhöht das Gold um 100, entfernt alle eigenen Einheiten und der nächste Spieler ist am Zug)
Cheat 2: Zahle 50 Gold, entferne eine Farm von jedem Gegner, nächster am Zug
Beispiel: CWTC

Leave Game wurde durch Leave Lobby ersetzt, da eine Lobby nur ein Spiel hostet!

5 Verbindungsunterbruch

Clientseitiger Umgang Der Nutzer wird darauf hingewiesen, dass der Server nicht erreichbar war und auf den connect to server screen weitergeleitet.

Serverseitiger Umgang Im Servermenü, werden die anderen Clients in diesem Menü über die Änderung mittels SET Kommando informiert. In der Lobby das gleiche und wenn die Lobby leer wäre, so wird diese gelöscht. In dem Spiel bleibt der werden alle Gebäude und Felder des Spielers gelöscht (So als hätte er das Spiel verlassen) erhalten und er wird einfach übersprungen. Nur noch ein Spieler: Gewinnt automatisch! Kein Spieler: Spiel Ende!