Praktikum Verteilte Systeme – WS2013/14

Aufgabe 1

Gruppe 2, Team 4
Erwin Lang, Leon Fausten

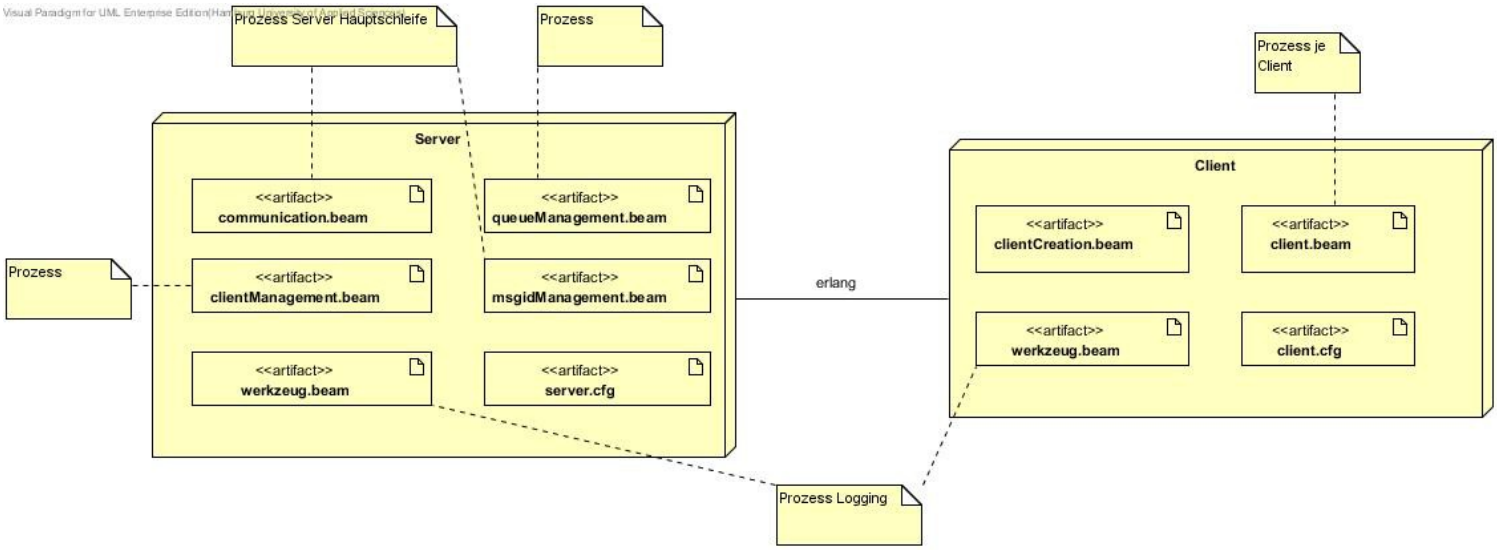
Allgemein

Die Aufgabe wurde in einem Erlangprojekt realisiert. Server und Client befinden sich im selben Projekt um den Test über die Konsole aus IntelliJ heraus zu erleichtern. Die entsprechenden Module sind:

VS_Aufgabe1

- communication.erl
- clientManagement.erl
- msgidManagement.erl
- queueManagement.erl
- werkzeug.erl
- client.erl
- clientCreator.erl

Deployment



Prozesse

- Jeder gestartete Client läuft in seinem eigenen Prozess
- Die Hauptschleife des Servers (communication) und das msgidManagement laufen im gleichen Prozess
- Das queueManagement läuft in einem Prozess
- Das clientManagement läuft in einem Prozess
- Das logging läuft immer in einem eigenen Prozess

Alle Prozesse auf Serverseite werden durch die Hauptschleife initialisiert.

Server

Lebenszeit des Servers

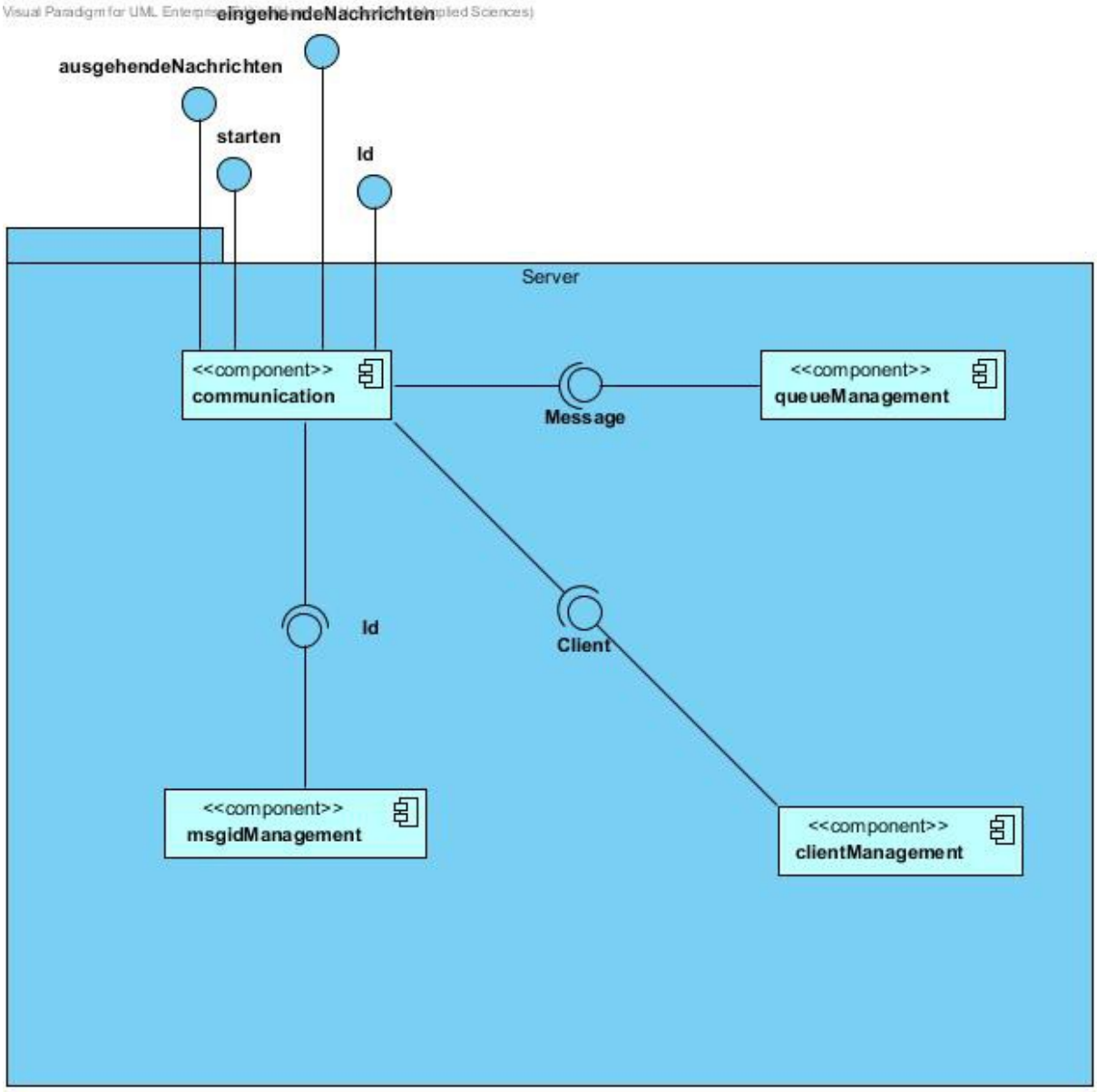
Die Lebenszeit des Servers wird innerhalb der server.cfg festgelegt.

Die angegebene Zeit legt fest, wie lange der Server inaktiv sein darf ohne neue Anfragen zu erhalten.

Bei jeder Anfragen an den Server wird der Timer neu gestartet.

Wenn der Timer abläuft werden alle gestarteten Server Prozesse beendet.

Komponenten



communication

Diese Komponente ist für die vollständige eingehende und ausgehende Kommunikation zuständigen.

Alle Anfragen gehen an diese: Anfrage einer neuen Nachrichten Id, Anfragen von Nachrichten und hinterlassen von Nachrichten.

Außerdem werden alle Nachrichten von hier aus an den Client gesendet: NachrichtenIds und neue Nachrichten.

Es werden alle Anfragen an die dafür zuständigen Komponenten weitergeleitet.

Innerhalb der communication-Komponente wird die Hauptschleife gestartet. In dieser Schleife werden alle von extern ankommenden Nachrichten verarbeitet. Ebenfalls werden hier Informationen über die aktuelle NachrichtenId und die Konfigurationen gespeichert, die aus der Datei geladen werden.

msgidManagement

Hier wird eine neue freie NachrichtenId ermittelt und an den anfragenden Client gesendet.

Da hier keine Werte gespeichert werden können, wird die aktuelle NachrichtenId in der Hauptschleife innerhalb von communication gespeichert.

Die Nachricht {getmsgid, self()} wird hier verarbeitet.

queueManagement

Hier wird die Holdbackqueue (HBQ) und die Deliveryqueue (DLQ) verwaltet. Sobald das erste Mal eine neue Nachricht angefragt wird oder eine Nachricht gespeichert werden soll wird ein neuer Prozess gestartet, der die Anfrage bearbeitet.

Die Anfragen {getmessage, Number} und {dropmessage, {Nachricht, Number}} werden hier verarbeitet. Die Übertragung der Nachrichten von HBQ zu DLQ wird hier ebenfalls vorgenommen.

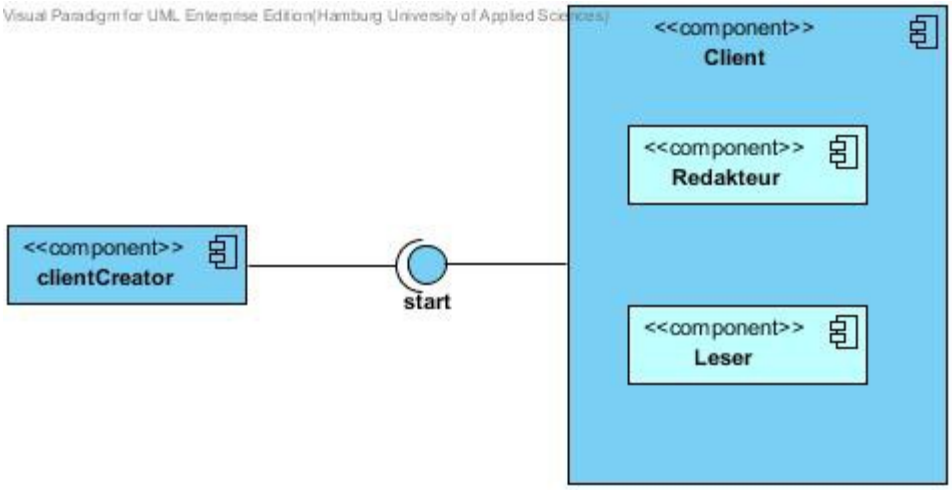
Die HBQ und die DLQ werden mithilfe der Schleife im Prozess gespeichert.

clientManagement

Die Clients werden in dieser Komponente verwaltet. Sobald das erste Mal Nachrichten angefragt werden wird ein neuer Prozess hierfür gestartet. Der Client wird hinzugefügt, sofern er noch nicht existiert, ansonsten wird nur sein Timer aktualisiert und die zuletzt angefragte Nachricht gespeichert.

Client

Komponenten



Client

Die Client-Komponente beinhaltet das laden der Konfiguration für den Client, sowie die in der Aufgabenstellung beschriebenen Abläufe von Redakteuren und Lesern.

Um den Wechsel von Redakteur zu Leser zu erleichtern, werden die Subkomponenten Redakteur und Leser beide im Modul *client.erl* implementiert.

clientCreator

Diese Komponente dient dazu eine bestimmte Anzahl an Clients zu erzeugen und zu starten um den Test mit mehreren Clients durchführen zu können. Dies ist der Einstiegspunkt der Clientseite. Der clientCreator erhält beim Start den Server als Parameter

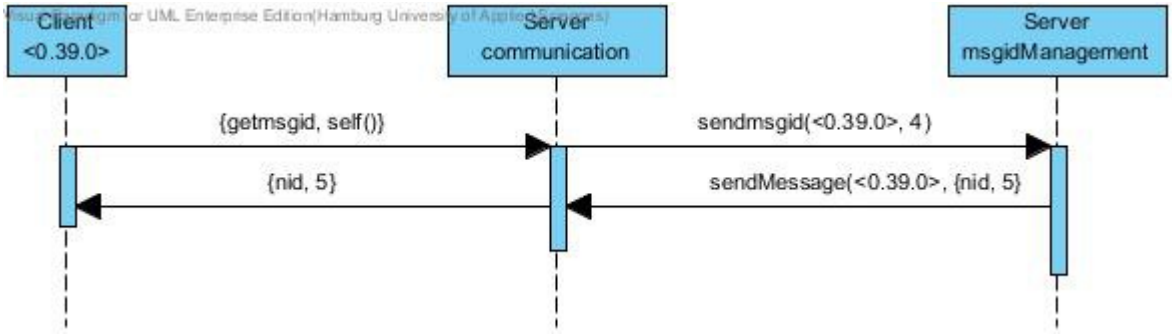
Extern

Werkzeug

Das beigelegte Werkzeug-Modul wird von sowohl Client als auch Server für das Logging, die Listenoperationen und den Timer verwendet. Das Logging wird jeweils immer in einen eigenen Prozess ausgelagert und blockiert somit nicht.

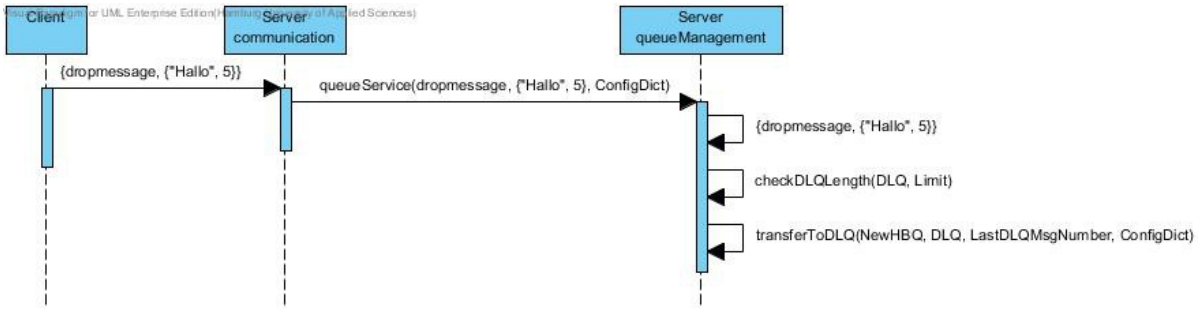
Sequenzdiagramme

Client fordert neue Nachrichtenid an



Die Client mit der ProzessId <0.39.0> sendet die Nachricht {getmsgid, self()}. Die communication empfängt diese Nachricht und führt die sendmsgid(PID, LastMsgId) Funktion der msgidManagement Komponente aus. Diese berechnet die neue Id und ruft mit dieser die sendMessage(PID, Nachricht) Funktion auf. Die schickt die Nachricht {nid, nextId} an den Client.

Client sendet Nachricht, Schwellwert wird erreicht



Wir gehen davon aus, dass der Server sich zuvor die NachrichtenId 5 wie im Diagramm zuvor geholt hat.

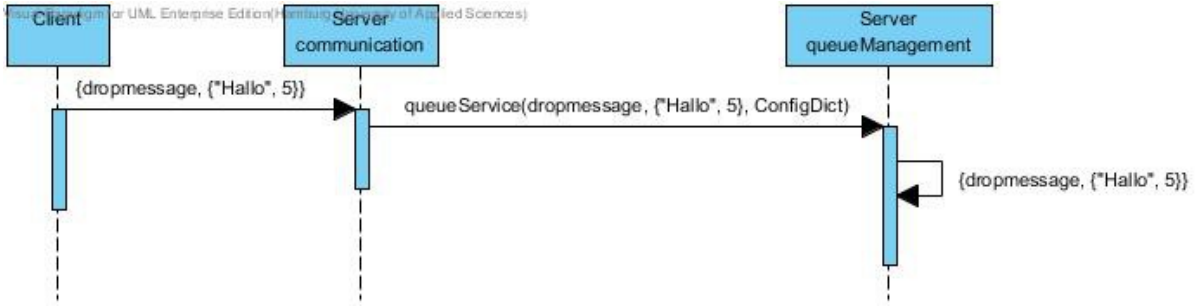
Der Client sendet die Nachricht {dropmessage, („Hallo“, 5}} an den Server. Der communication Prozess leitet die Anfrage an die queueManagement Komponente weiter. Diese startet den queueManagement Prozess falls nicht bereits zuvor geschehen. Anschließend wird an diesen Prozess die Nachricht {dropmessage, („Hallo“, 5}} weitergeleitet. Von hier läuft die Anfrage nebenläufig.

Der communication Prozess ist erneut bereit neue Nachrichten zu verarbeiten.

Der queueManagement Prozess fügt die Nachricht zur HBQ hinzu. Wenn die Bedingung anz(HBQ) > KAP(DLQ)/2 erfüllt wird, werden alle Nachrichten bis zur nächsten Lücke der HBQ in die DLQ transferiert. Wenn dabei eine Lücke entsteht wird eine Fehlernachricht hinzugefügt.

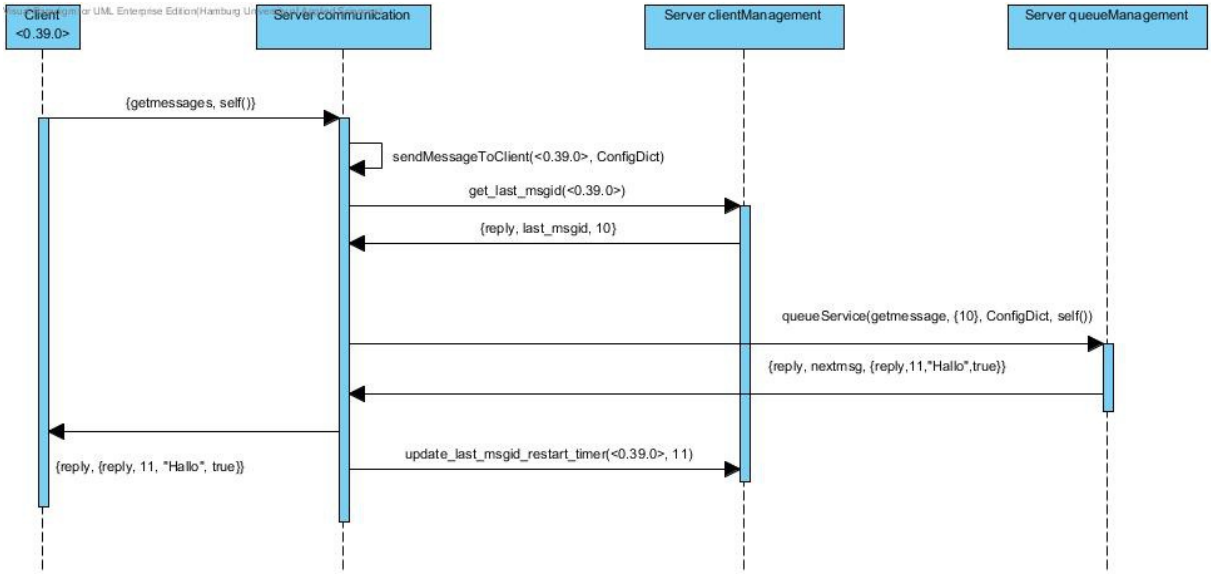
Vor dem Transfer jeder Nachricht wird jedoch die Größe der DLQ überprüft. Wenn die DLQ voll ist, werden solange Nachrichten vorne gelöscht bis die neuen Nachrichten transferiert werden können.

Client sendet Nachricht, Schwellwert wird nicht erreicht



Dies funktioniert wie Fall davor, nur das keine Nachricht in die DLQ übertragen wird.

Client fragt erfolgreich neue Nachricht an



In der DLQ steht eine Nachricht mit der Id 11. Folgenachrichten sind vorhanden.

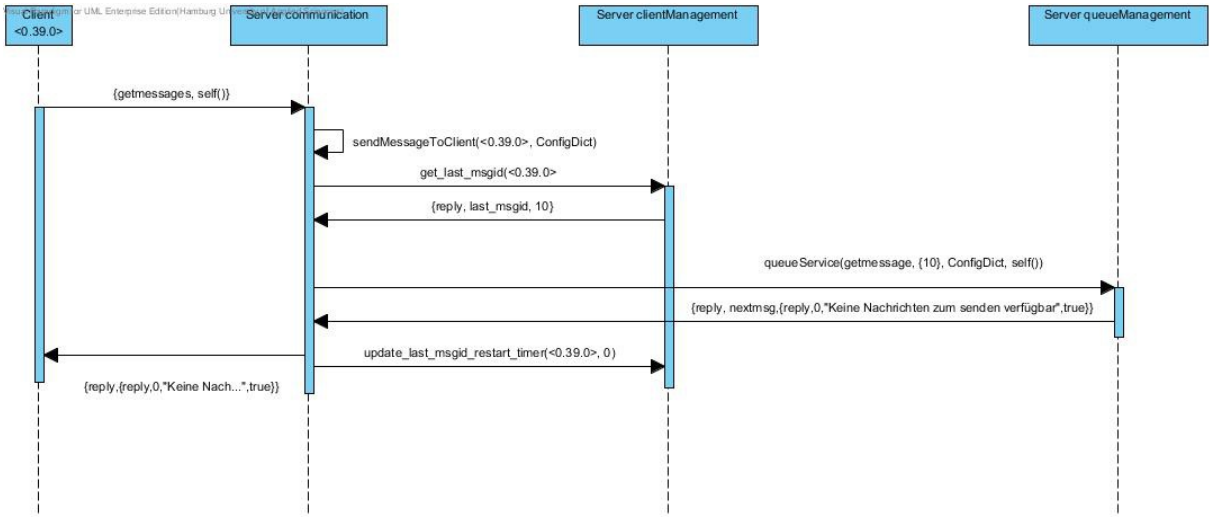
Der Client fordert neue Nachrichten an. Der communication Prozess fragt zunächst beim clientManagement nach welche NachrichtenId der Client zuletzt gelesen hat. Das clientManagement kennt den Client und gibt die NachrichtenId 10 zurück.

Nun wird bei dem queueManagement die nächste Nachricht angefragt. Die kleinste Nachricht in der DQL hat die Id 11 und wird zurückgegeben.

Diese Nachricht wird nun vom communication Prozess an den Client gesendet.

Als letztes teilt der communication Prozess dem clientManagement die neue gelesene NachrichtenId mit und der Timer um sich den Client zu merken wird neugestartet.

Client fragt ohne Erfolg neue Nachrichten an



Wie schon im Erfolgsfall beschrieben. Der Unterschied besteht darin, dass statt einer richtigen Nachricht eine Dummy-Nachricht geschickt wird.