[Datum]

Steffen Giersch &Maria Lüdemann

BAI 5 Verteilte Systeme bei Herrn Professor Klauck

HAW Hamburg

Entwurf

Verteilte Systeme Praktikum 1

Inhaltsverzeichnis

[Server 2](#_Toc400451939)

[Allgemeines 2](#_Toc400451940)

[Abblaufplan 2](#_Toc400451941)

[Pseudocode 2](#_Toc400451942)

[Client 3](#_Toc400451943)

[Allgemeines 3](#_Toc400451944)

[Ablaufplan 3](#_Toc400451945)

[Datentypen 3](#_Toc400451946)

[ADTs 3](#_Toc400451947)

[Deliveryqueue 3](#_Toc400451948)

[HoldBackQueue 4](#_Toc400451949)

[Logging 4](#_Toc400451950)

Entwurf Praktikumsaufgabe 1

# Server

## Allgemeines

Configparameter sind in einer server.cfg hinterlegt dort stehen:

Timeout -> Zeit des Servers die er wartet bevor er sich herunterfährt wenn keine Anfragen von Clienten mehr kommen

Deliveryqueuesize -> Die Länge die die Deliveryqueue haben darf

ClientTimeOut -> Die Zeit die sich der Server einen Clienten merkt bevor er ihn löscht wenn er sich nicht meldet.

## Abblaufplan

Siehe Anhang 1

## Datentypen

Nachricht = {Text,ClientOut,HBQIn,DLQIn,ClientIn}

Text -> String in dem die eigentliche Nachricht enthalten ist

ClientOut -> Timestamp von dem Moment wenn die Nachricht den Clienten(Redakteur) verlässt, bei Initialisierung 0

HBQIn -> Timestamp von dem Moment wenn die Nachricht die Holdbackqueue erreicht, bei Initialisierung 0

DLQInt -> Timestamp von dem Moment wenn die Nachricht die Deliveryqueue erreicht, bei Initialisierung 0

ClientIn -> Timestamp von dem Moment wenn die Nachricht den Clienten(Reader) erreicht, bei Initialisierung 0

Nummer = Integer

## ADTs

### Deliveryqueue

Eine Liste von Tupeln die jeweils zwei Elemente haben einmal die Nachricht (oben definiert) und die Nr als Integer der Nachricht.

DLQADT = [{Nachricht, Nr}]

createNew() :: void -> DLQ

Ein neues DLQ Objekt wird erstellt und zurückgegeben. Agier wie ein Konstruktor.

add(Msg, Nr, Queue) :: Nachricht x Nummer x DLQ -> DLQ

Die Nachricht wird der Queue hinten zugefügt so, dass sie sortiert bleibt. Wenn die Queue voll ist und eine weitere Nachricht kommt rein, wird die älteste Nachricht gelöscht.

get (Nr, Queue) :: Nummer x DLQ -> (Nachricht, Nummer , flag)

Gibt die Nachricht mit der angeforderten Nummer Nr zurück. Die Nummer ist die der erfolgreich angeforderten Nachricht. Das Atom „flag“ gibt an ob noch weitere Nachrichten existieren.

Wenn keine Nachricht mit der angeforderten Nummer in der DLQ existiert, dann wird nur das Atom „false“ zurückgegeben.

### HoldBackQueue

Eine Liste an Tupeln mit jeweils zwei Elementen eine Nachricht, Nachrichtdatentyp und Nr Integer

HBQADT = [{Nachricht, Nr}]

createNew() :: void -> HBQ

Ein neues HBQ Objekt wird erstellt und zurückgegeben. Agier wie ein Konstruktor.

add( Msg, Nr, Queue) :: Nachricht x Nummer x HBQ -> HBQ

Die Nachricht der Queue hinzugefügt und die Queue nach Nachrichtennummer aufsteigend sortiert. Wenn die Queue eine Größe erreicht die der Hälfte der Größe der DLQ entspricht werden Lücken zwangsweise geschlossen und die DLQ nachgefüllt.

### ClientList

Eine Liste an Tupeln die drei Elemente beinhalten

ClientList = [{ClientID, LastNumber, TimeStamp}]

createNew() :: void -> ClientList

add(ClientID, LastNumber, CurrentTime) :: Nummer x Nummer x TimeStamp -> ClientList

Fügt einen neuen Clienten mit seiner ID, der Nummer der Nachricht die er als letztes angefragt hat und der Zeit zu der er dies tat hinzu.

exist(ClientID) :: Nummer -> Boolean

Fragt die Liste ob es einen bestimmten Clienten schon gibt

update(CurrentTime) :: TimeStamp -> ClientList

Läuft über die Liste und prüft ob es Clienten gibt die sich länger als einen gewisser Intervall nicht gemeldet haben und löscht sie aus der Liste.

setTime(ClientID, CurrentTime) :: Nummer x TimeStamp -> ClientList

Erneuert die Zeit die zu einem Clienten gespeichtert wird wenn er sich meldet.

# Client

## Allgemeines

Aufgeteilt in Leser und Redakteur und darf nur aus einem Prozess bestehen.

Beide Clienten Teile liegen im selben Prozess sind aber autonom und laufen Sequentiell ab

In einer Config Datei client.cfg sind, Startparameter definiert und können dort für verschiedenen Testläufe verändert werden. Dort zufinden sind:

Timeout -> Wie lange läuft der Client bevor der sich terminiert

Minimale Zeit des Wartens -> 2 Sec

Start Wartezeit -> Initiale Zeit die zwischen den Nachrichten gewartet wird

### RedakteurClient

Fragt Nachrichtennummer vom Server ab und schreibt ihm mit der Nummer eine Antwort dies wird fünf mal wiederholt, dann holt sich der Client eine Nummer antwortet darauf aber nicht nochmal.

Danach wird der warte Intervall geändert und zum LeseClient gewechselt und ihm eine Liste der fünf Nachrichtennummern die er mindestens zu erwarten hat geschickt.

### LeseClient

Der LeseClient fragt den Server an ob es Nachrichten für ihn gibt und zeigt diese dann in seiner GUI an. Solange ihm der Server signalisiert, dass es noch neue Nachrichten gibt fragt der Client weiter nach. Erst wenn es keine mehr gibt wird zum Redakteur gewechselt. Der Leser überprüft ob er all die Nachrichten bekommt deren Nummer in der Liste stehen die er vom Redakteur bekommen hat und loggt ob sie alle da sind oder ob welche fehlen.

# Logging

## Server

Der Server nutzt einen Logging Prozess der in NServer.log schreibt

Serverstart:

Server Startzeit: 30.04 17:37:12,375| mit PID <0.870.0>

NachrichtenNummer an Clienten verschickt:

Server: Nachrichtennummer 5 an <9595.773.0> gesendet

Client fragt Nachricht an:

2-client@Brummpa-<0.771.0>-KLC: 45te\_Nachricht. C Out: 30.04 17:37:32,874|(45); HBQ In: 30.04 17:37:32,875| DLQ In:30.04 17:37:38,969|.(45)-getmessages von <9595.772.0>-false

Server fährt herunter:

Downtime: 30.04 17:38:40,719| vom Nachrichtenserver <0.870.0>; Anzahl Restnachrichten in der HBQ:5

### ADTs

HBQ loggt wenn sie Nachrichten transifiert

QVerwaltung1>>> Nachrichten [10,9,8,7,6,5,4,3,2,1] von HBQ in DLQ transferiert.

HBQ nimmt neue Nachricht auf

4-client@Brummpa-<0.773.0>-KLC: 5te\_Nachricht. C Out: 30.04 17:37:16,515|(5); HBQ In: 30.04 17:37:16,516|-dropmessage

HLQ schließt Lücken: BSP.

\*\*\*Fehlernachricht fuer Nachrichten 61 bis 70 um 30.04 17:37:54,328|.

DLQ löscht Nachrichten

QVerwaltung1>>> Nachrichten [6,5,4,3,2,1] von DLQ geloescht.

ClientList vergisst Clienten

Client <9595.772.0> wird vergessen! \*\*\*\*\*\*\*\*\*\*\*\*\*

## Client

Der Client nutzt einen Logging Prozess der in client\_<Nummer><Clienthost>.log schreibt.