Team: Falco Winkler (FW), Daniel Schruhl (DS)

Aufgabenteilung:

- DLQ (DS)
- HBQ (FW)

Quellenangaben:

Bearbeitungszeitraum:

- 26.03.2017 (FW,DS)
- 03.04.2017 (DS)
- 04.04.2017 (DS)

Aktueller Stand:

Änderung des Entwurfs:

- Formatierung angepasst
- Komponentendiagramm erweitert

Inhaltsverzeichnis

1 Einführung und Ziele

Es soll eine Message of the Day Anwendung erstellt werden. Dabei werden von verschiedenen Clients an einen Server verschiedene Nachrichten des Tages gesendet. Die Clients rufen vom Server alle Nachrichten ab, so dass jeder Client alle Nachrichten in einer festen Reihenfolge hat.

1.1 Randbedingungen

Es soll eine Client/Server-Architektur implementiert werden. Der Server verwaltet dabei die ihm von den Clients gesendeten Nachrichten. Das beinhaltet eine feste Numerierung der Nachrichten.
Die Clients rufen dabei in bestimmten Abständen die Nachrichten ab. Falls ein Client dem Server schon bekannt ist, bekommt der nur die ihm noch unbekannten (neuen) Nachrichten.
Der Server muss sich also die Clients merken. Es soll mit einer Holdbackqueue und einer Deliveryqueue gearbeitet werden, um die korrekte Auslieferung in einer bestimmten Reihenfolge der Nachrichten zu garantieren.

1.2 Kontextbegrenzung

Das System soll in Erlang umgesetzt werden. Es muss auf Computern mit Linux Betriebssystem lauffähig sein.

2 Gesamtsystem

2.1 Kontextsicht

2.2 Bausteinsicht

Das Softwareprodukt besteht aus mehreren Modulen und Paketen (Abbildung 1). Diese Pakete setzen sich zusammen aus dem Client-Paket und dem Server-Paket.

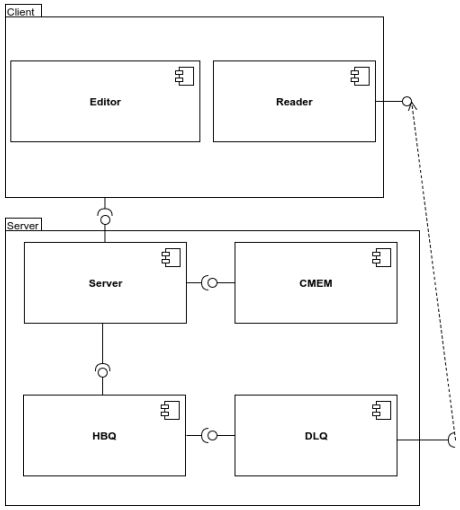

Abbildung 1: Komponentendiagramm der Message Of The Day App

Das Server Paket beinhaltet das Server-Modul und alle vom Server-Modul verwendeten Datenstrukturen.
Das Server-Modul ist für alle Funktionalitäten des Servers zuständig. Dazu gehört das Nummerieren und Verwalten der Nachrichten und die Verwaltung der Clients. Das Server-Modul benutzt daher per Schnittstelle das HBQ-Modul und das CMEM-Modul. Das Server-Modul stellt eine Schnittstelle für die Clients bereit.

Das CMEM-Modul ist für die Speicherung der Clients und ihrer aktuellen zu erwartenden Nachrichten Nummer zuständig. Das CMEM-Modul soll als lokale ADT realisiert werden. Diese wird nur vom Server angesprochen.

Das HBQ-Modul ist für die Holdbackqueue zuständig und regelt die Sortierung der einkommenden Nachrichten in die Deliveryqueue und der damit verbundenen Fehlerbehandlung. Dabei ist das HBQ-Modul als entfernte ADT realisiert. Diese wird nur von dem Server-Modul verwendet.

Das DLQ-Modul realsiert die Deliveryqueue, die für die Auslieferung der Nachrichten in Reihenfolge an die Clients zuständig ist. Die Schnittstelle des DLQ-Moduls wird nur vom HBQ-Modul konsumiert.

Das Client-Paket beinhaltet die Client-Module. Diese sind zum einen der Lese Client (Reader-Modul) und der Redakteur Client (Editor-Modul). Beide Clients sind als ein Prozess implementiert und verwenden die vom Server bereitgestellte Schnittstelle. Der Redakteur Client ist für das Schicken von Nachrichten zuständig und der Lese Client für das Lesen von Nachrichten.

2.3 Laufzeitsicht

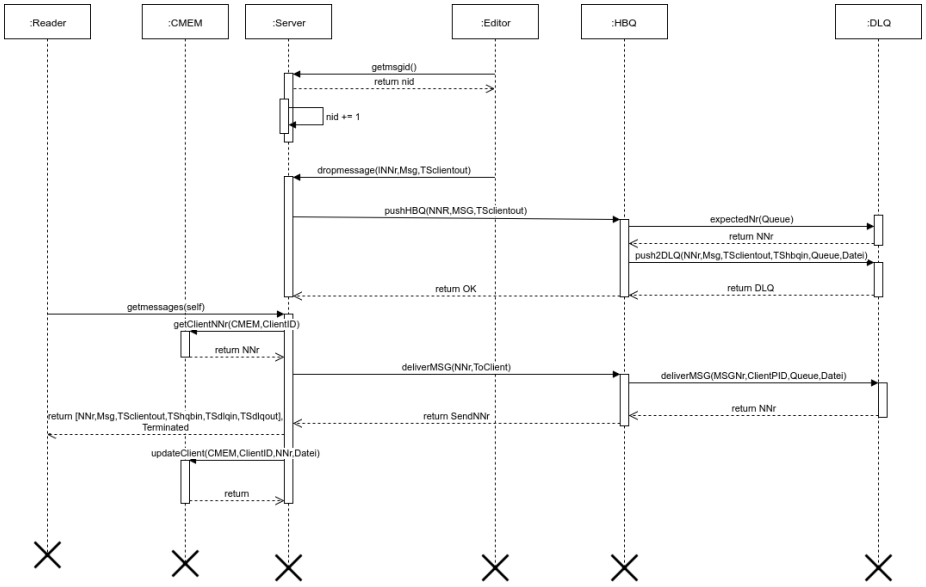

Abbildung 2: Sequenzdiagramm bei fehlerfreiem Nachrichtenaustausch

2.4 Verteilungssicht

3 Subsysteme und Komponenten

3.1 Server

3.1.1 Aufgabe und Verantwortung

Der Server hat die Aufgaben

1. Sich anfragende Clients mit ProzessID und Zeitstempel zu merken
2. Clients die ihnen zugehörigen Nachrichten zurückzugeben

Für die erste Aufgabe wird das Modul CMEM verwendet. Zum Senden an die Clients leitet der Server die Nachrichten zunächst an die HBQ weiter. Erscheint eine Anfrage vom Client, erfolgt über die HBQ der Aufruf an die DLQ, die Nachricht zu übermitteln.

3.1.2 Entwurfsentscheidungen

Der Server bearbeitet Anfragen per message passing. Die NNr ist ein Zählerwert, der nach jedem Herausgeben an einen Client inkrementiert wird.

3.1.3 Außensicht

Es werden Schnittstellen zur Abfage der aktuellen NNr, zum Holen & Speichern von Nachrichten bereitgestellt. Nach einer bestimmten, aber nach Außen unbekannten Zeit werden auch bereits abgeholte Nachrichten ausgeliefert.

3.1.4 Innensicht

Anfragen werden in einer Endlosschleife abgefragt und nur an die jeweiligen Komponenten weitergeleitet. Fehlerbehandlung erfolgt in den jeweiligen Komponenten. Bei jeder Anfrage erfolgt die Abfrage & Aktualisierung der CMEM, um sich den anfragenden Client zu merken und Anfragen an die HBQ korrekt zu stellen.

3.1.5 Konfigurationsparameter

1. Latenzzeit bestimmt die maximale Zeit die der Server ungenutzt läuft.
2. Lebenszeit des Clients bestimmt die Zeit, für die sich der Server einen Client merkt.
3. Name definiert den Namen der Server - Node
4. HBQ - Name & Node definieren die Verknüpfung zur HBQ - Node

3.1.6 Benutzungsschnittstelle

Der Server wird nur durch seine Schnitstellen vom Client verwendet.

3.2 CMEM-Modul

3.2.1 Aufgabe und Verantwortung

Dieses Modul hat die Aufgabe sich anfragende Clients mittels einer ProzessID und eines Zeitstempels zu merken, und diese für den Server bereitzustellen.

3.2.2 Entwurfsentscheidungen

Als Datenstruktur wird ein Dictionary verwendet. Jeder Client wird mittels seiner ProzessID eindeutig auf einen Offset und auf einen Zeitstempel der letzten Anfrage abgebildet. Der Offset wird verwendet, um sich zu merken, welche Nachricht als letztes an den Client gesendet wurde.

3.2.3 Außensicht

Die CMEM wird nur über Funktionsaufrufe durch den Server aufgerufen.

3.2.4 Innensicht

Intern arbeiten die Methoden deleteIdleClients und insertOrUpdateClient auf der Datenstruktur. getTimestampAndOffset gibt für einen Client den Entsprechenden Offset und Timestamp zurück. Nur der Server verarbeitet die zurückgegebenen Werte.

3.2.5 Konfigurationsparameter

3.3 Komponente HBQ

3.3.1 Aufgabe und Verantwortung der HBQ

Von Redakteur-Clients gesendete Nachrichten werden hier zwischengespeichert. Gibt es keine Lücke in der fortlaufenden Nummerierung der Nachrichten, werden Sie an die DLQ weitergeleitet. Bei einer Überfüllung der HBQ (2/3 der Maximalen Kapazität der DLQ) wird die älteste Lücke in den Nachrichten mit einer künstlichen Nachricht geschlossen, und es werden erneut alle Lückenlosen Nachrichten an die DLQ weitergeleitet. Die HBQ stellt nur ein Interface für den Server bereit.

3.3.2 Entwurfsentscheidungen

Es wird die Aktuelle NNr gespeichert. Jede Nachricht, die der aktuellen erwarteten NNr entspricht, wird direkt weitergeleitet. Wenn noch auf alte Nachrichten gewartet wird, wird die nachricht in die HBQ eingefügt. Wenn die Nachrichtennummer größer als der erwarteten Nachrichtennummer ist, müssen alle bereits empfangenen-nachrichten ab der erwarteten Nachrichtennummer durchgegangen werden. Die Nachrichten, die aufeinanderfolgende Sequenznummern haben werden an die DLQ ausgeliefert.
if currentNNr == NNr currentNNr +=1 pushToDLQ(NNr) else insert(NNr) currentNNr = pushAllConsecutiveFrom(currentNNr)

3.3.3 Außensicht

3.3.4 Innensicht

3.3.5 Konfigurationsparameter

- Die Maximalgröße der DLQ wird verwendet um die HBQ bei kritischer Überfüllung zu leeren.

3.4 DLQ-Modul

3.4.1 Aufgabe und Verantwortung

Die Deliveryqueue hat die Aufgabe Nachrichten an Clients zuzustellen und stellt eine Datenstruktur dar, die eine maximale Menge (Kapazität) an Nachrichten hält. Sie verhält sich dabei wie von einer Warteschlange zu erwarten beim Einfügen und Herausholen von Nachrichten (FIFO). Nur das HBQ-Modul darf auf das DLQ-Modul zugreifen. Das DLQ-Modul sendet über eine Schnittstelle der Clients die Nachrichten an die Clients.

3.4.2 Schnittstelle

```
/* Initialisieren der DLQ */
initDLQ(Size,Datei): Integer X Atom -> DQueue

/* Abfrage welche Nachrichtennummer in der DLQ gespeichert werden kann */
expectedNr(Queue) : DQueue -> Integer

/* Speichern einer Nachricht in der DLQ */
push2DLQ([NNr,Msg,TSclientout,TShbqin],Queue,Datei) :
    MSG_list X DQueue X Atom -> DQueue

/* Ausliefern einer Nachricht an einen Leser-Client */
deliverMSG(MSGNr,ClientPID,Queue,Datei):
    Integer X PID X DQueue X Atom -> Integer
```

initDLQ(Size,Datei): initialisiert die DLQ mit Kapazität Size. Bei Erfolg wird eine leere DLQ zurück geliefert. Datei kann für ein logging genutzt werden.

delDLQ(Queue): löscht die DLQ. Bei Erfolg wird ok zurück geliefert.

expectedNr(Queue): liefert die Nachrichtennummer, die als nächstes in der DLQ gespeichert werden kann. Bei leerer DLQ ist dies 1.

push2DLQ([NNr,Msg,TSclientout,TShbqin],Queue,Datei): speichert die Nachricht [NNr,Msg,TSclientout,TShbqin] in der DLQ Queue und fügt ihr einen Eingangszeitstempel an (einmal an die Nachricht Msg und als expliziten Zeitstempel TSdlqin mit erlang:now() an die Liste an. Bei Erfolg wird die modifizierte DLQ zurück geliefert. Datei kann für ein logging genutzt werden.

deliverMSG(MSGNr,ClientPID,Queue,Datei): sendet die Nachricht MSGNr an den Leser-Client ClientPID. Dabei wird ein Ausgangszeitstempel TSdlqout mit erlang:now() an das Ende der Nachrichtenliste angefügt. Sollte die Nachrichtennummer nicht mehr vorhanden sein, wird die nächst größere in der DLQ vorhandene Nachricht gesendet. Bei Erfolg wird die tatsächlich gesendete Nachrichtennummer zurück ge-

liefert. Datei kann für ein logging genutzt werden.

3.4.3 Entwurfsentscheidungen

Die Deliveryqueue wird als Liste realisiert. Das erste Element der Liste ist dabei eine Liste der Nachrichten und das zweite Element der Liste gibt die Kapazität der DLQ an.

```
/* Nachrichten Format */
/* minimal 3 Elemente, pro Station kommt eins hinzu; maximal 6 Elemente */
MSG_List := [NNr,Msg,TSclientout,TShbqin,TSdlqin,TSdlqout]:
    [Integer X String X 3-Tupel X 3-Tupel X 3-Tupel X 3-Tupel]

/* DLQ Format */
/* Nachrichtenliste ist eine Liste aus MSG_List und hat */
/* maximal Kapazitaet Anzahl an Elementen */
DLQ := [Nachrichtenliste, Kapazitaet]: [List X Integer]
```

Neue Nachrichten werden am Anfang der Liste angefügt. Das vereinfacht die Nachfrage nach der nächsten zu speichernden Nachrichtennummer in der DLQ. Demnach werden Nachrichten am Ende der Liste entnommen. Wenn eine Nachricht in die DLQ eingetragen wird, wird ans Ende der Nachricht der aktuelle Zeitstempel zum Zeitpunkt des Einfügens hinzugefügt.
Nachrichten werde der DLQ entnommen, wenn die Größe der Liste gleich ihrer Kapazität ist und eine neue Nachricht hinzugefügt werden soll.

Beim senden einer Nachricht durch Angabe der Nachrichtennummer wird in der DLQ nach der Nachricht mit der Nachrichtennummer gesucht. Wenn diese gefunden wird, wird die Nachricht an den Client (mit der ClientPID) gesendet und die Nachrichtennummer zurück gegeben. Falls die Nachricht nicht gefunden werden konnte, wird die nächste Nachricht mit der nächst größeren Nachrichtennummer an den Client gesendet.

3.4.4 Konfigurationsparameter

- Kapazität der DLQ