Team: Team 13, Krisztina Gyarmati und Hendrik Sieling

Aufgabenaufteilung:

1. Der Entwurf wurde gemeinsam erstellt.

Quellenangaben:
https://pub.informatik.haw-hamburg.de/home/pub/prof/klauck_christoph/VS/BA_Erlang.pdf
http://users.informatik.haw-hamburg.de/~klauck/VerteilteSysteme/Aufgaben.pdf
http://users.informatik.haw-hamburg.de/~klauck/VerteilteSysteme/aufg1-rmi.html

Begründung für Codeübernahme: -

Bearbeitungszeitraum:
Aufgabenstellung verstehen, analysieren: ca. 7 Stunden
Entwurf fertigstellen: ca. 5 Stunden

Aktueller Stand: Entwurf fertig.

Änderungen im Entwurf: <Vor dem Praktikum auszufüllen: Welche Änderungen sind bzgl. dem Vorabentwurf vorgenommen worden.>

Entwurf:

Aufgabenstellung:
Implementation einer Client/Server-Anwendung. Der Server verwaltet die Nachrichten, die von unterschiedlichen Clients zugesendet wurden. Die Clients fragen in bestimmten Abständen die aktuellen Nachrichten ab. Der Server erinnert sich daran, welche Nachrichten er an welche Clients zuletzt zugestellt hat. Nach einer gewisse inaktive Zeit werden die Lese-Clients vergessen.

- Die Ausgaben sind in die Dateien Server@<Node>.log,
  HB-DLQ@<Node>.log und
  client_<Nummer><Node>.log (z.B. Client_2client@KI-VS.log)

zu schreiben.

- Sourcedateien und eine Readme.txt müssen mitgeliefert werden
- Programmiersprache: Erlang/OTP

Client

- Beim Starten des Clients ist max. 1 Parameter erlaubt.
- Darf maximal aus einem Prozess bestehen (neben dem Prozess zum loggen).
- Bei der Initialisierung müssen die folgende Werte (die im server.cfg angegeben sind) gesetzt werden

```
{clients, 5}.                 % Client Anzahl
{lifetime, 42}.               % Client Lebenszeit
{servername, wk}.             % Name des Servers
{servernode, 'server@KI-VS'}. % Node des Servers
{sendeintervall, 3}.          % Intervall zwischen 2 Sendereignissen
```

Redakteur-Client

- Sendet in bestimmten Abständen eine Textzeile in der Form:

Rechnername (z.B. lab18), Praktikumsgruppe (z.B. 1) und Teamnummer (z.B. 03), also "lab18103" beinhalten und aktuelle Systemzeit (die der Sendezeit entsprechen soll) und ggf. anderen Text, zum Beispiel

```
0-client@lab18-<0.1313.0>-C-1-03: 22te_Nachricht. Sendezeit: 16.05
18:01:30,769|(22);
```

- Der Abstand wird nach 5 Textzeilen zufällig um ca. 50% vermindert oder erhöht, darf aber nicht kleiner als 2 Sekunden sein.
- Der Redakteur fragt nach 5 gesendeten Textzeilen nach eine neue Nachrichtennummer, sendet aber keine Nachricht mit dieser sechsten Nummer (dieses Ereignis muss auch im log vermerkt sein).

Leser-Client

- Fragt alle Textzeilen von dem Server ab und stellt sie in seiner GUI dar. Pro Anfrage bekommt er genau eine noch unbekannte Textzeile. In einer Flag steht ob es noch weitere unbekannte Nachrichten für ihn gibt.

● Falls es keine neue oder überhaupt keine lieferbare Nachrichten gibt, bekommt der Client eine Dummy-Nachricht.
● Der Client merkt sich die Nummern die er vom Server erhalten hat und und fügt einer als Leser-Client erhaltenen Nachricht, die durch sein Redakteur-Client erstellt wurde, die Zeichenfolge `*******` an, etwa

```
6te_Nachricht. C Out: 11.11 21:12:58,720|(6); HBQ In: 11.11 21:12:58,720| DLQ
In:11.11 21:13:01,880|.*******; C In: 11.11 21:13:07,190|
```

Server

Die Aufgabe des Servers ist den Nachrichtenaustausch zwischen den (Redakteur- und Leser-) Clients zu regeln.
Er verwaltet die Nummervergabe der Nachrichtenzeilen.
● Bei der Initialisierung müssen die folgende Werte (die im server.cfg angegeben sind) gesetzt werden

```
{latency, 42}.                    % Server Lebenszeit
{clientlifetime,4}.               % Client Timeout
{servername, wk}.                 % Name des Servers
{hbqname, hbq}.                   % Name der HBQ
{hbqnode, 'hbqNode@Brummpa'}.     % Node der HBQ
{dlqlimit, 42}.                   % Maximale Größe der DLQ
```

● Wenn ein Client eine Nachricht senden will muss er zuerst eine Nachrichtennummer anfordern.
● Die Nummerierung fängt bei jedem Serverneustart bei 1 an und wird nach jede Abfrage um 1 erhöht. Die Nummer einer Nachricht ist auch seine eindeutige ID.
● Der Server merkt nicht von wem die Nachricht gesendet wurde und schaut auch nicht in die Nachricht hinein.
● Der Server fügt am Ende der Nachricht einen Zeitstempel hinzu, wenn die Nachricht in eine der beiden Queues eingefügt wird.
● Der Server speichert im CMEM wann die Lese-Clients zuletzt eine Abfrage gemacht haben und welche Nachricht sie zuletzt bekommen haben. Nach dem Ablauf des Client-Timeouts werden die Clients vergessen und bei dem nächsten Abfrage wie ein unbekannter LeseClient behandelt.
● Der Server terminiert sich, wenn die Zeit, die seit dem letzten Client-Abfrage vergangen ist größer ist als seine Lebenszeit.
● Der Server ist unter einem Namen <name> im lokalen Namensdienst von Erlang zu registrieren (`register(<name>,ServerPid)`).
● Darf maximal aus einem Prozess bestehen (neben dem Prozess zum loggen).
● Verwendete ADTs: HBQ (hbq.erl), DLQ (dlq.erl) und CMEM (cmem.erl).

ADTs

● Dürfen nur als Erlang-Liste implementiert werden. Tupeln als Hilfsstrukturen sind erlaubt.
● Werden beim Serverstart initialisiert.

HBQ

● Holdbackqueue, hier stehen die Nachrichten, die nicht an die Clients ausgeliefert werden können.

● Wenn mehr als ⅔-tel der echten Nachrichten im HBQ stehen muss die Lücke zwischen HBQ und DLQ geschlossen werden. Das passiert mit genau einer Fehlernachricht z.B.

```
Fehlernachricht fuer Nachrichtennummern 11 bis 17 um 16.05 18:01:30,580|.
```

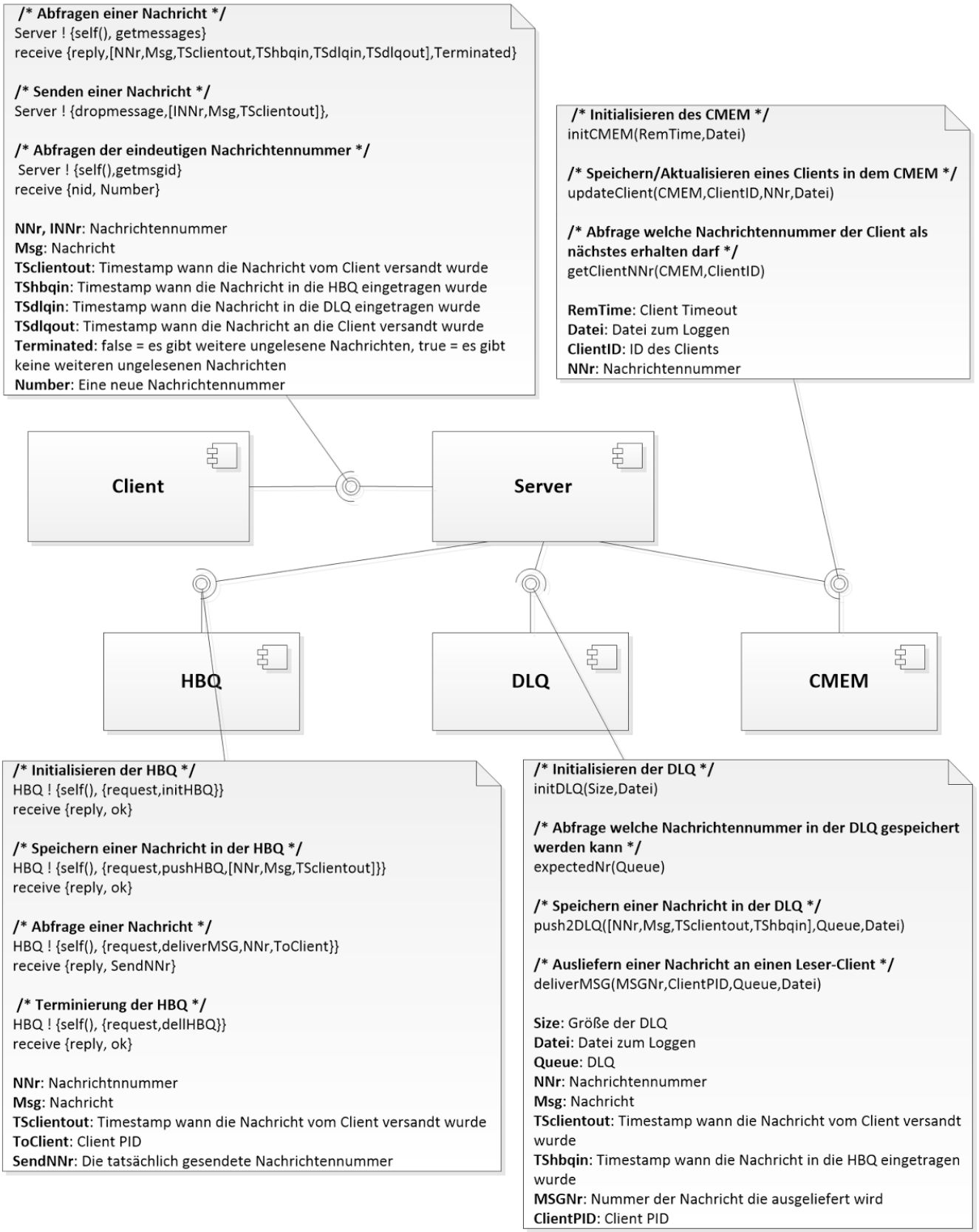
DLQ

● Deliveryqueue, hier stehen die Nachrichten, die an die Clients ausgeliefert werden können.
● Hat eine feste Größe.

CMEM

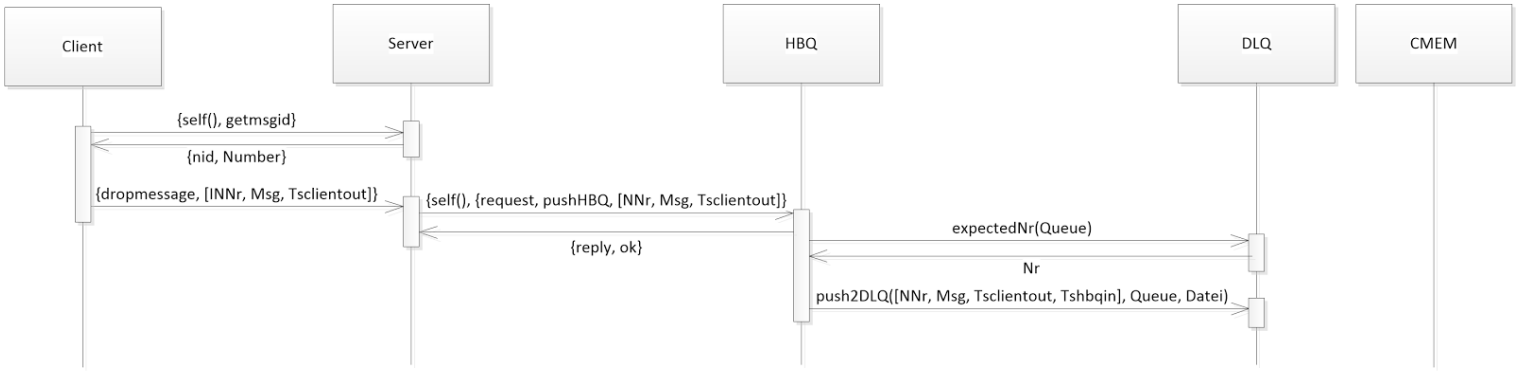
● Gedächtnis für die Leser-Clients.

Komponenten und Schnittstellen



Sequenzdiagramme

Client sendet eine Nachricht



Client fragt eine neue Nachricht ab