**Team**: 4, Dino Buskulic, Milena Dreier

**Aufgabenaufteilung**:

1. Alles wurde gemeinsam erarbeitet

**Quellenangaben**: Praktikumsaufgabe

**Aktueller Stand**: Sequenzdiagramm für den Ablauf der Kommunikation zwischen Server und Editor, Server und Reader.Zusätzlich ein Sequenzdiagramm für die Kommunikation innherhalb des Servers. Komponentendiagramm für Server und Client.

**Entwurf**:

**Funktionalität Server:**

Der Server holt sich beim Start seine Parameter über die eingebundene .cfg Datei und setzt somit seinen **Namen**, **Lebenszeit**, **Lebenszeit des Clients** (wie lange der Client in der Liste bekannt sein soll), die **Maximale Größe der DLQ.**

**Schnittstellen:**

Für die Kommunikation mit dem Client bietet der Server folgende (fest vorgegebene) **Schnittstellen** an:

- getmsgid → Reserviert dem Client eine Nummer für die nächste Nachricht
- dropmessage → Bekommt vom Client eine Nachricht
- getmessages → Anfrage der aktuellen Nachricht in der DLQ vom Client

Die eigentliche Funktionalität besteht darin, intern die Nachrichten, die vom Client erhalten werden in zwei verschiedenen Queues zu verwalten. Die Nachrichten die ankommen, werden erst in die **HBQ** (***H****old****B****ack****Q****ueue*) aufgenommen.

Anschließend wird geprüft ob die Nachricht in die **DLQ** (***D****elivery****Q****ueue*) verschoben werden kann oder nicht. Zusätzlich soll der Server bekannte Clients verwalten. Wenn ein Client zum ersten Mal eine Anfrage stellt, wird er in eine Liste eingetragen, zusätzlich merkt sich der Server den letzten Wert (**Nachrichtennummer**) den der Client erhalten hat. Wenn der Client bei einer Anfrage bereits bekannt ist, wird der Wert aus der Liste geholt.

**ADT´s:**

Für beide Queues werden entweder sortierte Listen benutzt oder Dictionarys (wird beim Implementieren geprüft) .Für die Liste der Clients bietet sich auch das Dictionary an. Dadurch kann die PID des Clients und die Nummer der letzten abgelieferten Nachricht gespeichert werden.

**HBQ →Sortierte Liste**

**DLQ** →Sortierte Liste

**Clients** → Dictionary mit Key Value Paar

## Timouts prüfen

**ServerTimeout** (Beenden des Servers): Ein eigener Timer prüft ob die Differenz der letzten Client Anfrage zur Systemzeit länger als die vorgegebene Wartezeit ist und beendet in dem Fall den Server.

**ClientTimout** (Vergessen der Clients): Ein Timer prüft alle x (Wert aus .cfg File) Sekunden ob die Differenz des Zeitstempels und des in der Liste hinterlegten Zeitstempels für den Client den Wert x überschritten hat.

## Fehlernachricht erstellen

Wenn nach dem einfügen einer Nachricht in die HBQ die maximale Größe der HBQ erreicht ist, wird eine Fehlertextzeile erstellt.

## Funktionalität Client:

Der Client wird mit einem Parameter gestartet, der den **Namen des Servers** festlegt. Zusätzlich holt sich der Client aus der eingebetteten .cfg Datei die Parameter für die Werte **Anzahl der zu startenden Clients**, der **Lebenszeit eines Clients** und dem **Zeitintervall für das warten in einem Sendevorgang.**

Der Client hat zunächst 2 Zustände:

- Redakteur-Zustand (Editor)
- Lese-Zustand (Reader)

Zu Anfang, startet der Client im Redakteur-Zustand. Dort führt er seinen Sendevorgang aus und ladet Nachriten beim Server ab. Nachdem er den den Sendevorgang im Loop abgeschlossen hat, berechnet er seinen TimeoutValue neu. Der Wert wird per Zufall, entweder um 50% reduziert oder aber erhöht. Jedoch wird niemals die Untergrenze (1 Sekunde) unterschritten. Nach dem Sendevorgang im Redakteur-Zustand, werden die Nachrichtennummern gespeichert die verschickt wurden, um später entscheiden zu können welche Nachrichten tatsächlich beim Server gelandet sind und welche verloren gegangen sind. Zusätzlich soll der Redakteur-Client nach jedem Sendezyklus (5 x Senden) sich zwar eine Nummer vom Server holen, die Nachricht aber nicht abschicken.

## ADT´s

**Messagenumbermanager**: Liste für die Verwaltung der eigenen Nachrichtennummern

## Timeouts prüfen

**ClientTimeout** (Lebenszeit des Clients)**:** Ein Timer der nach der im .cfg Datei hinterlegten Lebenszeit den Client terminiert.