Team: 05, Eugen Deutsch, Ralf von der Reith
Aufgabenaufteilung:
1. Ralf: Entwurf Server und Nachrichtenverwaltung (HBQ, DLQ)
2. Eugen: Entwurf Client

Quellenangaben:
Bearbeitungszeitraum: -/-
Aktueller Stand: -/-
Änderungen des Entwurfs: -/-
Entwurf:

Server:

//TODO:
- Sequenzdiagramme für alle Aufrufe der Schnittstellen
- Funktionen des Nummerndienstes
- Komponentendiagramm ähnlich dem aus Klaucks Foliensatz.

Schnittstelle zu Clients

Die Schnittstelle zu den Clients stellt die Funktionalität des Servers nach außen in Form von Remote Procedure Calls zur Verfügung.

Die Schnittstelle kümmert sich ausschließlich um die Annahme von Anfragen und deligiert diese an die entsprechenden Komponenten weiter.

Folgende Anfragen werden vom Server angenommen:

getmessages:
Format: {SenderPID, getmessages}
SenderPID: PID des Senders
Funktion: Sendet die entsprechend der durch den Client zuletzt erhaltenen Nachrichten nächste Nachricht an den Client.

Der Server schaut nach, welche Nachricht zuletzt an den Client gesendet wurde und sendet dann die Nachricht mit der nächsthöheren Nummer an den Client zurück.
Ist ein Client dem Server noch nicht bekannt, wird die älteste verfügbare Nachricht versandt.
Sind keine neuen Nachrichten verfügbar, wird eine Dummy-Nachricht mit Terminated = true an den Client gesendet.

Die Antwort hat folgendes Format:
{reply, [NNr, Msg, TSclientout, TShbqin, TSdlqin, TSdlqout], Terminated}
NNr: Nummer der versendeten Nachricht
Msg: Text der Nachricht
TSclientout: Zeitstempel bei Versand der Nachricht durch den Client
TShbqin: Zeitstempel bei Empfang in HBQ
TSdlqin: Zeitstempel bei Einreihung in DLQ
TSdlqout: Zeitstempel bei Versand an den Client
Terminated: false -> es gibt noch weitere Nachrichten
true -> keine weiteren neuen Nachrichten

dropmessage

Format: {SenderPID, dropmessage, [NNr, Msg, TSclientout]}
SenderPID: PID des Senders
NNr: Nummer der Nachricht
Msg: Text der Nachricht
TSclientout: Zeitstempel bei Versand der Nachricht durch den Client
Funktion: Hinterlegt eine Nachricht auf dem Server.

Die Nachricht wird zur Speicherung an die Nachrichtenverwaltung delegiert. Details siehe Nachrichtenverwaltung.

Antwort des Servers:
Format: {reply, ok}

getmsgid
Format: {SenderPID, getmsgid}
SenderPID: PID des Senders
Funktion: Sendet die nächst-verfügbare eindeutige Nachrichtennummer an den Client

Antwort des Servers:
Format: {nid, NNr}
NNr: Nachrichten-ID

Andere:
Alle Nachrichten die keinem der oben genannten Formate entsprechen, werden ohne Rückmeldung verworfen.

Nummern-Dienst (ND)

Der Nummerndienst vergibt eindeutige Nachrichtennummern an die Clients in aufsteigender Reihenfolge.

//Funktionen?

Client-Memory (CMEM)

Im CMEM wird für jeden Client hinterlegt, welche Nachricht er zuletzt empfangen hat. Stellt ein Client für eine bestimmte Zeit keine weiteren Anfragen, wird er vergessen, d.h. sein entsprechender Eintrag gelöscht.

Funktionen:
initCMEM(RemTime,Datei) → CMEM
RemTime: Zeit in Sekunden, nach der ein Client vergessen wird.
Datei: Datei, die für Logging genutzt werden kann.
CMEM: ein leerer CMEM

Initialisiert den Client-Memory und gibt einen leeren CMEM zurück.

delCMEM(CMEM) → ok
CMEM: der zu löschendeCMEM

Löscht den CMEM und gibt ok zurück.

updateClient(CMEM,ClientID,NNr,Datei) → NeuerCMEM
CMEM: der aktuelle CMEM
ClientID: ID des Clients, der aktualisiert werden soll
NNr: die Nummer der zuletzt an den Client gesendeten Nachricht
Datei: Datei, die für Logging genutzt werden kann.
NeuerCMEM: der aktualisierte CMEM

Speichert bzw. aktualisiert den Eintrag des Clients ClientID mit der neuen Nachrichtennummer.

getClientNNr(CMEM,ClientID) → NNr
CMEM: der aktuelle CMEM
ClientID: ID des angefragten Clients
NNr: Nächste Nummer, die der Client erhalten darf.

Gibt die nächst-erwartete Nachrichtennummer für diesen Client zurück.

Nachrichten-Verwaltung:

Holdback-Queue (HBQ)

Die Holdback-Queue nimmt alle Nachrichten entgegen und gibt diese an die DLQ weiter, sofern dabei die lückenlose Durchnummerierung in der DLQ gewahrt bleibt.
Fehlen Nachrichtennummern, werden sämtliche Nachrichten mit höherer Nummer zurückgehalten, bis die Lücke gefüllt ist. Sobald die HBQ eine Größe von 2/3 der Maximalgröße der DLQ erreicht, wird diese Lücke durch eine Fehlernachricht ersetzt, welche die letzte Nummer der fehlenden Nachrichten als Nachrichtennummer erhält. Der Inhalt dieser Fehlernachricht gibt Auskunft darüber, welche Nachrichten ersetzt wurden.

Nachrichten aus der HBQ dürfen nicht an die Clients weitergegegeben werden.

Folgende Nachrichten werden durch die Nachrichtenverwaltung verarbeitet:

initHBQ
Format: {SenderPID, {request, initHBQ}}
SenderPID: PID des Senders

Initialisiert die HBQ und DLQ. Nach Initialisierung ist die HBQ leer.

Antwort der HBQ
Format: {reply, ok}

pushHBQ
Format: {SenderPID, {request, pushHBQ, [NNr, Msg, TSclientout]}
SenderPID: PID des Senders
NNr: Nummer der erhaltenen Nachricht.
Msg: Inhalt der erhaltenen Nachricht.
TSclientout: Zeitstempel bei Versand der Nachricht durch den Client.

Fügt der Nachricht einen Zeitstempel bei Eintragung in die HBQ an und hinterlegt diese

Nachricht in der HBQ.

Antwort der HBQ:
Format: {reply, ok}

DeliverMSG
Format: {SenderPID, {request, deliverMSG, NNr, ToClient}}
SenderPID: PID des Senders
NNr: Nummer der zu sendenden Nachricht
ToClient: PID des Clients, an den die Nachricht gesendet werden soll

Sendet die Nachricht mit der Nummer NNr an den Client ToClient. Ist die Nachricht mit der Nummer NNr nicht mehr in der DLQ hinterlegt, wird die älteste Nachricht versendet.

Antwort der HBQ:
Format: {reply, SendNNr}
SendNNr: Nummer der versendeten Nachricht

dellHBQ
Format: {SenderPID, {request, dellHBQ}}

terminiert den Prozess der HBQ.

Antwort der HBQ:
Format: {reply, ok}

Delivery-Queue (DLQ)

Die Delivery-Queue enthält alle Nachrichten, die derzeit versendet werden können. Sie hat eine feste Maximalgröße. Wird beim Speichern einer Nachricht in der DLQ die Maximalgröße überschritten, wird die älteste Nachricht verworfen.

Die DLQ kann nicht direkt, sondern nur über die HBQ angesprochen werden.

Funktionen:
initDLQ(Size, Datei) → DLQ
Size: Größe der zu erstellenden DLQ
Datei: Datei, die für Logging genutzt werden kann
DLQ: eine leere DLQ

Erstellt eine leere DLQ mit fester Maximalgröße und gibt diese zurück.

delDLQ(Queue) → ok
Queue: die zu löschende DLQ

Löscht die DLQ und gibt ok zurück.

expectedNr(Queue) → NNr
Queue: die aktuelle DLQ
NNr: die als nächstes erwartete Nachrichtennummer.

Gibt die als nächstes erwartete Nachrichtennummer zurück.

push2DLQ([NNr, Msg, TSclientout, TShbqin], Queue, Datei) → NeueDLQ
NNr: Nummer der zu speichernden Nachricht
Msg: Inhalt der zu speichernden Nachricht
TSclientout: Zeitstempel bei Versand der Nachricht durch den Client
TShbqin: Zeitstempel bei Speichern der Nachricht in der HBQ
Queue: aktuelle DLQ
Datei: Datei, die für Logging verwendet werden kann.
NeueDLQ: die modifizierte DLQ inklusive der zu speichernden Nachricht

Speichert die Nachricht in die DLQ und hängt einen Zeitstempel für den Eingang der Nachricht in die DLQ an.

deliverMSG(MSGNr, ClientPID, Queue, Datei) → SentNNr
MSGNr: Nummer der zu versendenden Nachricht
ClientPID: PID des Clients, an den die Nachricht versendet werden soll
Queue: aktuelle DLQ
Datei: Datei, die für Logging verwendet werden kann
SentNNr: Nummer der tatsächlich versendeten Nachricht.

Sendet die Nachricht mit der Nummer MSGNr an den Client mit der PID ClientPID. Ist diese nicht verfügbar, wird die älteste Nachricht der DLQ versendet.

Client: