Team: 08, Dieter Pisarewski, Maxim Rjabenko

Aufgabenaufteilung:

Maxim Rjabenko und Dieter Pisarewski:
Gemeinsame Entwurf und Implementierung

Quellenangaben:

http://www.erlang.org/doc/

Begründung für Codeübernahme:

Der größte Anteil vom Servercode wurde aus dem Projekt des vorherigen Semesters übernommen, das von Dieter Pisarewski implementiert wurde.

Bearbeitungszeitraum:

04.10.2013, 4 Stunden

10.10.2013, 8 Stunden

11.10.2013, 4 Stunden

14.10.2013, 4 Stunden

Aktueller Stand:

Server und Client sind implementiert, aber nicht auf mehreren Rechnern getestet.

Änderungen im Entwurf: -

Entwurf:

Client

Der Client versendet Nachrichten an den Server und holt sich die neuesten Nachrichten wieder ab.

Architektur

- Der Client wird logisch in den Redakteur- und in den Leseclient unterteilt.
- Die beiden logischen Clients wechseln sich sequentiell in der Ausführung ab.
- Mit kurzen Zeitabständen wird eine fortlaufende Nummer vom Server angefragt und eine entsprechende Nachricht versendet.
- Der Client merkt sich die für den Versand verwendete Nachrichtennummern.
- Die Nachricht enthält den Rechnernamen, die Praktikumsgruppe, Teamnummer sowie eine Systemzeit.
- Die Zeitabstände zwischen dem Versenden von Nachrichten werden nach 5 versandten Nachrichten geändert. Sie vergrößern oder verkleinern sich zufällig um 50%, mindestens jedoch um 1 Sekunde, dürfen aber nicht kleiner als 2 Sekunde sein.
- Das Versenden jeder sechsten Nachricht wird vergessen.

- Der Client empfängt alle neuen Nachrichten vom Server, bis dieser keine weitere Nachricht mehr ankündigt.
- Die Nachrichten werden bei der Ausgabe zusätzlich mit einem Zeitstempel versehen.
- Eigene Nachrichten werden mit ******* gekennzeichnet.
- Nach Ablauf einer eingestellten Lebenszeit wird der Client beendet.

Konfiguration

Folgende Werte sind konfigurierbar:

- clients, die Anzahl der zu startenden Clients.
- lifetime, die Zeit, nachdem sich der Client von selbst beendet.
- servername, der Name des Servers, zu dem sich der Client verbindet.
- sendeintervall, die Zeit in Millisekunden, nach der sich der Client beendet.
- anzahl_nachrichten, die Anzahl der Nachrichten, die hintereinander gesendet werden.

Realisierung

Wir haben den Clientcode in folgende Module unterteilt:

- client – die Zentrale Einheit des Clients mit einem Prozess.
- client_writer – enthält Funktionen zum Erstellen und Senden neuer Nachrichten an den Server(Redakteurclient).
- client_reader – enthält Funktionen zum Auslesen aller Nachrichten vom Server(Leseclient).

Die beiden logischen Einheiten, der Redakteur- und der Leseclient wechseln sich sequentiell ab, sodass das Senden und Abfragen der Nachrichten nacheinander erfolgt.

Der Programmablauf kann daher in einem einzelnen Prozess erfolgen und die oben genannten Punkte Schritt für Schritt abarbeiten.

Die Umsetzung des Clients erfolgt analog zur Architekturbeschreibung. Für das normale Versenden der Nachrichten wird deliver_messages aufgerufen und Nachrichten werden versendet. Für den Empfang der Nachrichten wird eine rekursive Funktion gestartet. Sie beendet sich, sobald der Server keine weiteren Nachrichten in der Queue meldet.

Server

Der Server verwaltet die Nachrichten, die von Clients gesendet werden und liefert diese in richtiger Reihenfolge den Clients zurück aus.

Architektur

- Der Server liefert dem Client auf Anfrage eine fortlaufende Nummer für die Nachricht.
- Die dem Server zugesendeten Nachrichten werden über Holdbackqueue und Deliveryqueue verwaltet, um die Reihenfolge einzuhalten.
- Der Server protokolliert am Ende der Nachricht den Eingangszeitstempel beim Eintritt in die Holdback- und Deliveryqueue.

- Die Größe der Deliveryqueue ist beschränkt, und alte Nachrichten werden bei Überschreitung der Größe entfernt.
- Die Holdbackqueue wird benötigt, weil die Nachrichten nicht in Reihenfolge eintreffen können und eine Lücke entstehen kann.
- Besteht zwischen der ersten Nachricht der Holdbackqueue und der letzten Nachricht der Deliveryqueue keine Lücke, wird die Nachricht aus der Holdbackqueue in die Deliveryqueue verschoben. Dies erfolgt, solange die Liste nicht leer ist und keine Lücke entsteht.
- Überschreitet die Größe der Holdbackqueue die Hälfte der Maximalgröße für die Deliveryqueue, besteht eine Lücke. Die wird durch eine Fehlernachricht, die direkt in die Deliveryqueue geschoben wird und die Nummer der ersten Nachricht der Holdbackqueue minus 1 trägt, geschlossen und die Holdbackqueue wird weiter abgearbeitet.
- Erfolgt in einer vorgegebenen Zeit keine weitere Nachrichtenabfrage von einem Client, terminiert sich der Server.

Konfiguration

Folgende Werte sind konfigurierbar:

- latency, die Zeit, bis der Server sich nach der letzten Abfrage beendet.
- clientlifetime, die Zeit, nach der ein Client vergessen wird.
- servername, an dem die Nachrichten gesendet werden.
- dlq_limit, die Maximalgröße der Deliveryqueue.
- log_file, der Name der Logdatei für die Programmausgaben.

Realisierung

Wir haben den Servercode in folgende Module unterteilt:

- server – die zentrale Einheit des Servers mit dem Prozess, der Anfragen von Clients empfängt.
- sender – enthält Funktionen zum Senden von Nachrichten an Clients.
- queue_manager – enthält Funktionen zum Verwalten von Holdbackqueue und Deliveryqueue.
- client_manager – ein Prozess, der Daten über Clients speichert.
- tools – enthält allgemeine Hilfsfunktionen.

Der Server besteht aus 2 Prozessen: Server und ClientManager.

Die zentrale Einheit (der **Server** selber) übernimmt die Kommunikation mit den Clients und die Verarbeitung der drei in der Schnittstelle beschriebenen Nachrichten.

- Auf getmsgid wird eine fortlaufende Zahl zurückgeliefert. Dies erledigt der Server selbst, ohne auf die anderen Einheiten zuzugreifen.
- Auf getmessages wird die älteste ungelesene Nachricht in der Deliveryqueue an den Client gesendet.

- Auf dropmessage wird die Nachricht in der Holdbackqueue gespeichert. Anschließend werden Nachrichten aus der Holdbackqueue in die Deliveryqueue übertragen.

Jede erhaltene Nachricht kommt zuerst in die Holdbackqueue. Nach dem Speichern der Nachricht in die Holdbackqueue wird zunächst die auf die neueste Nachricht aus der Deliveryqueue folgende Nachricht gesucht. In diesem Fall wird diese von der Holdbackqueue in die Deliveryqueue geschoben. Ist die Deliveryqueue nun größer als die festgelegte Maximalgröße, wird außerdem die älteste Nachricht gelöscht. Diese Suche wird wiederholt, bis die Holdbackqueue leer ist, oder eine Lücke gefunden wird.

Anschließend wird die Größe der Holdbackqueue geprüft. Überschreitet diese die Hälfte der Maximalgröße der Deliveryqueue, geht der Server davon aus, dass in der Holdbackqueue eine Lücke entstanden ist. Dann füllt er diese Lücke mit einer Fehlernachricht und schiebt sie direkt in die Deliveryqueue mit der größten Nachrichtennummer in der Lücke.

Bei der Auslieferung der Nachrichten an Clients fragt der Server die letzte Nachrichtennummer, die dem Client gesendet wurde, beim ClientManager ab.

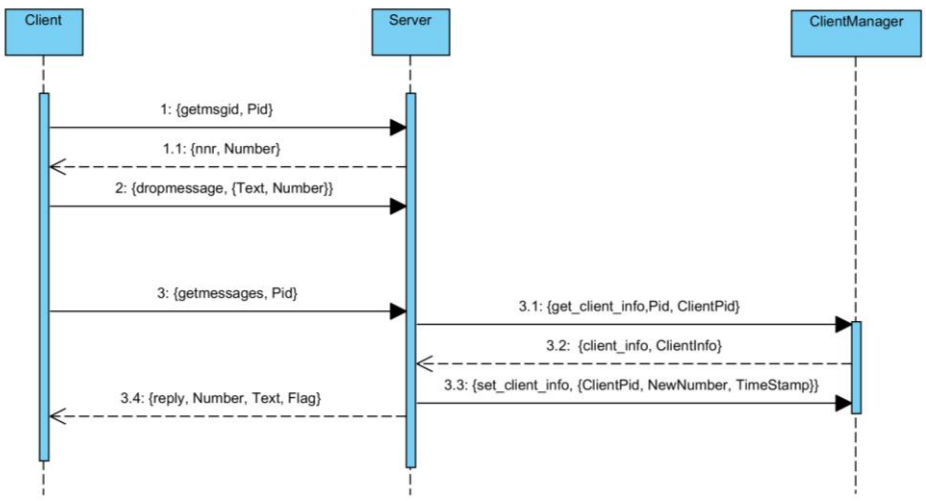
Dabei überprüft der Server, ob der Client dort schon existiert. In diesem Fall prüft er den gespeicherten Zeitstempel und setzt die letzte Nachrichtennummer des Clients zurück, wenn die vergangene Zeit vom Zeitstempel bis zur aktuellen Zeit ein in Konfiguration vorgegebenes Zeitintervall überschreitet. Danach wird der Zeitstempel des Clients aktualisiert und die Nachrichtennummer auf die kleinste Nachrichtennummer in der Deliveryqueue gesetzt.

Als nächstes holt sich der Server Nachrichten aus der Deliveryqueue, wählt die benötigte Nachricht aus und sendet sie an den Client.

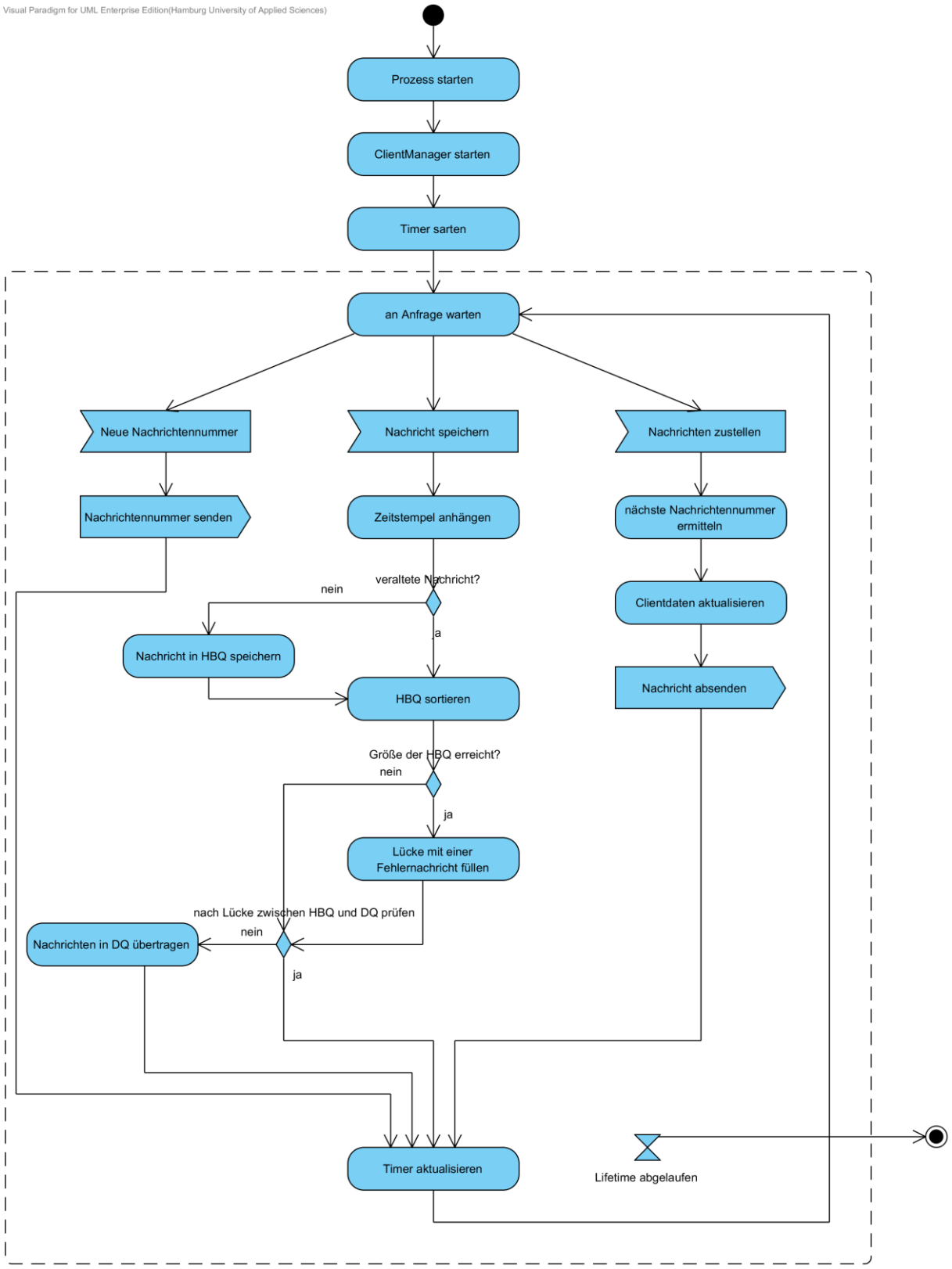
Ein weiterer Prozess des Servers ist der **ClientManager.** Er verwaltet eine Liste von registrierten Clients mit zugehörigen Informationen (ClientPid, letzte gesendete Nachrichtennummer, letzte Zugriffszeit).

Ein Client landet in dieser Liste, wenn er Nachrichten vom Server anfordert. Bevor er dort gespeichert wird, prüft der Server, der die oben genannte Aktion initiiert, ob der Client dort schon existiert

Kommunikationsdiagramm



Aktivitätsdiagramm - Server



Aktivitätsdiagramm - Client