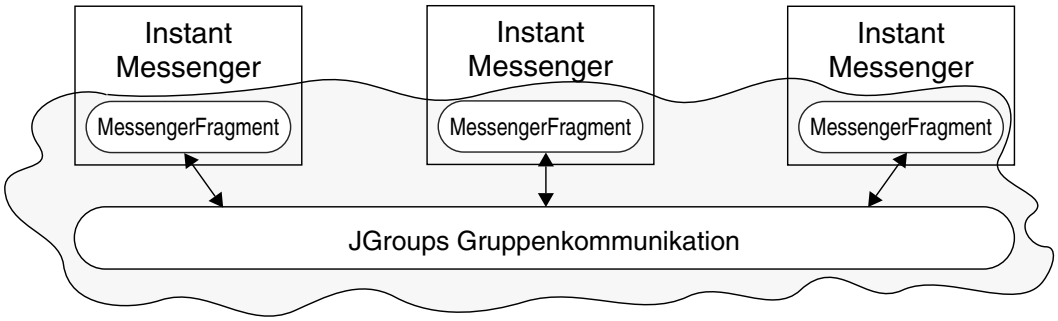
Übungsaufgabe #5: FORMI - Instant Messenger

18.12.2008

In dieser Aufgabe soll eine verteilte und fehlertolerante Instant Messenger Anwendung erstellt werden. Hierfür soll die von FORMI (Fragmentierte Objekte basierend auf RMI) und JGroups (Gruppenkommunikation) zur Verfügung gestellte Funktionalität genutzt werden. Die notwendigen Quellen und Bibliotheken befinden sich unter `/proj/i4mw/pub/aufgabe5/` bzw. `/local/formi` und `/local/JGroups/`. (Für die Arbeit am eigenen Rechner sind die entsprechenden Pakete in diesen Verzeichnissen zu verwenden.)



a) Um den Instant-Messenger Dienst zu initialisieren bzw. um diesem beitreten zu können, soll die Klasse `InstantMessenger` erstellt werden. Sie soll das (lokale) Fragment aufnehmen und auch das Frontend für die Ein- und Ausgabe bereitstellen.
Erstellen Sie die Basisklassen `MessengerFragmentInterface` und `MessengerFragment` für den verteilten Dienst. Der Fragmentzustand enthält die Historie aller Nachrichten seit der Entstehung des ersten Fragments und soll durch die Klasse `MessengerState` realisiert werden. Um Nachrichten empfangen zu können, soll zusätzlich `ReceiverInterface` und `Receiver` implementiert werden, die dann als Callback an das Fragment übergeben werden können.

Folgende Methoden sollen demnach mindestens definiert werden:

```
(MessengerFragment)
public void sendMessage(String s);
public void setReceiver(ReceiverInterface receiver);
(Receiver)
public void receiveMessage(String msg);
(MessengerState)
public String[] getHistory();
```

Zusätzlich soll ein Makefile oder Antfile den FORMI-Compiler `(/local/formi/lib/compiler.jar)` aufrufen, damit die benötigten Fragmentschnittstelle generiert wird.

b) FORMI arbeitet wie RMI mit Remote-Referenzen. Üblicherweise werden diese ebenfalls verteilt verwaltet. Um die Aufgabe etwas zu vereinfachen, soll eine einzelne (globale) RMI-Registry verwendet werden. Sie soll auf dem Rechner des ersten Teilnehmers gestartet werden. (Tipp: Auf die nötigen Rechte im Sicherheitsmanager achten. Am besten alles erlauben!)

Beim Start des Instant Messenger soll überprüft werden, ob bereits ein Dienst (Fragment) vorhanden ist. Ist dies nicht der Fall, soll mittels `FragmentedObjectFactory` (Tipp: Standard Factory-Implementierung ist: `DefaultFragImplFactory`) ein neues Fragment erzeugt und an der RMI-Registry gebunden werden, andernfalls soll die vorhandene Remotereferenz genutzt werden um eine lokale Kopie des Fragments zu erhalten. Wichtig: Die Factory verwendet nicht den Standard-Konstruktor, sondern `FormiFragment(FragImplFactory fragImplFactory, Object[] cc)`, welcher entsprechend in MessengerFragment überladen werden sollte.

c) Um den Dienst fehlertolerant zu machen, soll zur Kommunikation JGroups verwendet werden (Eine ausführliche Dokumentation findet sich unter `http://www.jgroups.org`). Innerhalb des Fragments soll hierfür ein `JChannel` erzeugt werden, die notwendigen Kanal-Parameter finden sich auch in der Vorgabe.

In den Parametern können zusätzlich alle anderen Kommunikationsteilnehmer angegeben werden um die Verfügbarkeit zu gewährleisten. Zu Testzwecken kann zunächst mit statischen Parametern gearbeitet werden: Es wird nur die Kontaktadresse des ersten/initialen Fragments in die Konfiguration übernommen.

Nachdem das Fragment als Empfänger gesetzt wurde, kann es dem Kanal mit einer eindeutigen Nummer (z.B. der GUID - `fragImplFactory.guid.toString()`) beitreten.

Ereignisse auf dem Kommunikationskanal werden von JGroups wie folgt signalisiert:
`public void viewAccepted(View newView)`
Änderung an der Menge der Kommunikationsteilnehmer, darüber lassen sich die Kanalparameter dynamisch aktualisieren
`public byte[] getState()`
Der Objektzustand (des Fragments) wird durch einen anderen Teilnehmer angefordert
`public void setState(byte[] arg0)`
Den Objektzustand von einem anderen Teilnehmer anfordern
`public void receive(Message arg0)`
Eine Nachricht wurde empfangen

d) Bei der Übertragung der Remotereferenz wird (ausschließlich) die Fragment-Factory (`DefaultFragImplFactory`) serialisiert. Damit die neuen Adressen bei der Übertragung einer Referenz auch verfügbar sind, müssen sie in dem Factory-Objekt eines jeden aktiven Fragments abgelegt werden. Zu diesem Zweck können der Factory Kommunikationsparameter übergeben werden. Sie werden bei der Erzeugung initial durch

```
createObject(Class<? extends FormiFragment> objType,
    Class<? extends FragImplFactory> fragImplFac, GUID group,
    Object[] commCred, Object[] addFactoryArgs, Object[] addFragArgs)
```

gesetzt und später durch

```
public Object[] getCommunicationProperties();
public void setCommunicationProperties(Object[] communicationProperties);
```

verändert.

Die Kommunikationsparameter sollten also bei jeder Änderung aktualisiert werden, hierfür können die Statusmeldungen von JGroups verwendet werden.

Tipp: Für die Kommunikationsadressen kann `IpAddress` von JGroups verwendet werden. Allerdings ist diese Klasse nicht serialisierbar und muss entsprechend abgeleitet werden (z.B. in `SIpAddress`) bevor es an das Factory-Objekt übergeben wird.

e) Zuletzt soll nun noch die eigentlich verteilte Eigenschaft des Instant-Messenger-Dienstes umgesetzt werden: Die verteilte Nachrichtenhistorie.

Hierfür soll das Frontend von InstantMessenger so erweitert werden, dass die Historie ausgegeben werden kann. Der Aufbau der Historie wird durch das Messenger-Fragment geleistet: Es protokolliert sämtliche ein- und ausgehenden Nachrichten mit und hinterlegt sie in `MessengerState` (JGroups gewährleistet eine zuverlässige Kommunikation, die Historie muss daher nicht noch zusätzlich synchronisiert werden). Da sich die Historie nicht in dem Factory-Objekt befindet, muss sie von jedem Fragment zunächst angefordert werden. Dafür sollen die JGroup Methoden `getState` und `setState` entsprechend umgesetzt werden.

Bearbeitung: bis spätestens zum 15.01.2009/13:45 Uhr
Alle notwendigen Quelldateien müssen im SVN-Repository eingecheckt sein.