Universität Dortmund
Lehrstuhl Verteilte Systeme

Dr. Michael Zapf

Michael.Zapf@cs.uni-dortmund.de
Michael.Zapf@uni-kassel.de

vkiv@ls4.cs.uni-dortmund.de

Übung zur Vorlesung „Verteilt-kooperative Informationsverarbeitung“ - SS 2007

Blatt 2

Ausgabe 23.04. - Abgabe 07.05.

Aufgabe 2.1 – 10 Punkte

Ein Virtual Home Environment [1] ist eine Unterstützung des Anwenders im UMTS-Netz, die ihm den Eindruck vermittelt, dass er auch in fremden Netzen alle seine persönlichen Dienste wiederfindet (er sich also „wie zu Hause fühlt“). Die Herausforderung besteht dabei darin, dass der Anwender auch über größere Distanzen hinweg mit möglicherweise komplexen Funktionen seiner Umgebung versorgt werden soll. Dies würde zu erheblichem Kommunikationsaufwand führen.

Wäre dies ein mögliches Einsatzgebiet für mobile Agenten? Diskutieren Sie

- mögliche Einsatzszenarien für mobile Agenten: Wofür könnte man die mobilen Agenten einsetzen?
- Voraussetzungen an Endgeräte und Infrastruktur: Funktioniert das auf meinem Java-Handy, oder muss der Provider mit spezieller Hardware zentral/pro Benutzer aushelfen?
- den Einsatz anderer Mechanismen für mobilen Code: Würde REV/CoD hier nicht ausreichen?

[1] UMTS Virtual Home Environment; http://www.umtsworld.com/technology/vhe.htm

Aufgabe 2.2 – 15 Punkte

Implementieren Sie in Java einen Tupelraum (mit der Standard-Java-API) und demonstrieren Sie die Funktion mittels eines einfachen Beispiels. Achten Sie darauf, dass ihre Tupelraum-Teilnehmer auf verschiedenen Rechnern laufen können. (Beachten Sie bitte die Hinweise zu Programmieraufgaben in der Übung.)

Aufgabe 2.3 – 10 Punkte

Warum kennt Java keine starke Migration? Versetzen Sie sich in die Rolle eines Entwicklers im Java Community Process, der solche Anfragen zuhauf bekommt und argumentieren Sie für oder gegen die Einführung von Mechanismen in Java, welche die Implementierung der starken Migration begünstigen. Wie wahrscheinlich halten Sie es, dass man Ihrem Wunsch als Agentensystementwickler nach solchen Mechanismen nachkommt, jetzt, da Java in einem Gemeinschaftsprozess weiterentwickelt wird (und nicht nur von Sun alleine gestaltet wird)? Suchen Sie ggf. auch im Developer Forum nach Hinweisen.

Aufgabe 2.4 – 15 Punkte

Im Folgenden sei ein mobiler Agent in (Pseudo-)Java dargestellt.

Wir gehen davon aus, dass die Migration stets gelingt und keine Fehlerbehandlung notwendig ist. Der „Itinerary“ ist die Reiseroute; mittels „getNextPlace“ bekommt man die nächste Station auf der Reise.

Mittels „doTask“ wird eine Aktion in der Aufgabenliste für die angegebene Stelle durchgeführt. Der Agent wird an irgendeiner Stelle gestartet, die nicht unbedingt gleich „MyHome“ ist. Mittels „migrate“ wechselt der Agent die Ausführungsumgebung.

Wir gehen von einer Standard-Java-VM aus. Die *migrate*-Methode unterbricht den Programmablauf und bewirkt eine Migration des Agenten zur genannten Stelle. Die Details der Namenssuche (also das Bestimmen der Position von Stellen im Netz) seien hier außer Acht gelassen.

```
public class WanderAgent extends MobileAgent {

   Location startplace = ... ; // zeigt auf "MyHome"

   public void start() {
      if (!thisPlace().equals(startplace)) migrate(startplace);
      TaskList tl = thisPlace().getTaskList();
      Itinerary it = tl.getItinerary();
      while (it.hasNext()) {
         Location next = it.getNextPlace();
         migrate(next);
         int state = tl.doTask(next);
         if (state==ABORT) break;    // verlasse "while"-Schleife
      }
      migrate(startplace);
      report();
   }
}
```

a) Beschreiben Sie das offenbar erwünschte und das tatsächliche Verhalten des Agenten.

b) Schreiben Sie das Programm für Java so um, dass es seinen Zweck erfüllt. (Sie können ebenso wie oben die Fehlerbehandlung auf das Notwendigste reduzieren.)

Hinweis: Die Methoden migrate, thisPlace, report sowie die Variable state und ABORT seien in der Elternklasse definiert. Die Objekte Location, Itinerary und TaskList sowie deren Methoden sind dem Agenten bekannt. Verwenden Sie Instanzvariablen (Felder) für Daten, die Ihr Agent bewahren soll und teilen Sie die Ausführung in Phasen ein, die in der start-Methode selektiert werden. import-Anweisungen oder die Implementierung der referenzierten Klassen sind nicht notwendig.