Profilbasierte Rekonfiguration kleiner Endgeräte

"Roboter unterstützte Sensornetzwerke"

Gerhard Fuchs, Sébastien Truchat, Falko Dressler
Informatik 7, Universität Erlangen-Nürnberg
Martensstr. 3, 91058 Erlangen
{gerhard.fuchs, sebastien.truchat, dressler}@informatik.uni-erlangen.de

Zusammenfassung: In diesem Paper wird ein Vorhaben beschrieben, bei dem Sensorknoten von mobilen, autonomen Robotern rekonfiguriert werden sollen. Hierzu wird ein allgemeiner, profilbasierter Mechanismus vorgestellt, der für eine PDA/WLAN-Anwendung entwickelt wurde. Dieser Mechanismus wird mit besonderem Bezug auf die speziellen Anforderungen und Besonderheiten im anvisierten Szenario angepasst. Besondere Beachtung liegt dabei auf der Leichtgewichtigkeit des Mechanismus und aller beteiligten Operationen.

1 Einführung

Die primären Forschungsaspekte der Autonomic Networking Gruppe des Lehrstuhls für Rechnernetze und Kommunikationssysteme sind das Entwickeln und Beherrschen von selbst-organisierenden, selbst-konfigurierenden, und adaptiven Methoden für Kommunikationssysteme und interagierende autonome Systeme. Diese Aspekte werden im Rahmen des ROSES[1]-Projekts (Robot assisted Sensor Networks) am Beispiel von mobilen Robotern und stationären Sensornetzen untersucht. Hierbei betrachten wir nicht nur die Teilsysteme (z.B. energiegewahre Aufgabenverteilung bei autonomen Robotern [3] oder bio-inspirierte Mechanismen zur Selbstorganisation von Sensornetzwerken [1]), sondern auch deren Kombination. Wir unterscheiden zwischen Sensornetzwerk unterstützten Roboterschwärmen, d.h., dass die Roboter das Sensornetzwerk z.B. zur präziseren Lokalisierung verwenden [2] und Roboter unterstützten Sensornetzen, d.h., dass die mobilen Roboter das Sensornetzwerks z.B. instand halten und warten.

Ein Beispiel einer Wartungsmaßnahme ist die profilbasierte Rekonfiguration der Sensorknoten unter Berücksichtigung des aktuellen Kontexts (momentane Aufgaben des Systems) durch die mobilen Roboter. Unser Ziel ist es, dieses Szenario mit dem im Mo.S.I.S-Projekt (Modulare Softwareentwicklung für Interoperative Systeme) erarbeiteten Mechanismus zu realisieren. Im Gegensatz zu den klassischen Client-Server-Ansätzen wie z.B. [5, 6] basiert die verwendete Architektur auf autonomen Knoten.

Das nachfolgende Paper gliedert sich wie folgt: Abschnitt 2, erklärt die theoretischen Grundlagen des Profiling Mechanismus, Abschnitt 3 die geplante praktische Umsetzung. Abschnitt 4 zeigt die bisherigen Erkenntnisse und die weitere Vorgehensweise.

2 Das Profiling Konzept

Im Rahmen des Mo.S.I.S. Projektes wurde an unserem Lehrstuhl ein Konzept für generische Rekonfiguration erarbeitet [7]. Ein Entwurfsmuster beschreibt einen Profiling Mechanismus, der im Zusammenhang mit einem leichtgewichtigem RPC Protokoll [8] eine

[1] Projekt-Homepage: http://www7.informatik.uni-erlangen.de/~dressler/projects/roses/roses_en.shtml

Plattformübergreifende "any-to-any" Rekonfiguration von mobilen Endgeräten ermöglicht (Abb. 1). Zielsetzung war es, einen sehr leichtgewichtigen Mechanismus zu entwickeln, um diesen auch für sehr kleine eingebettete Plattformen zu verwenden.

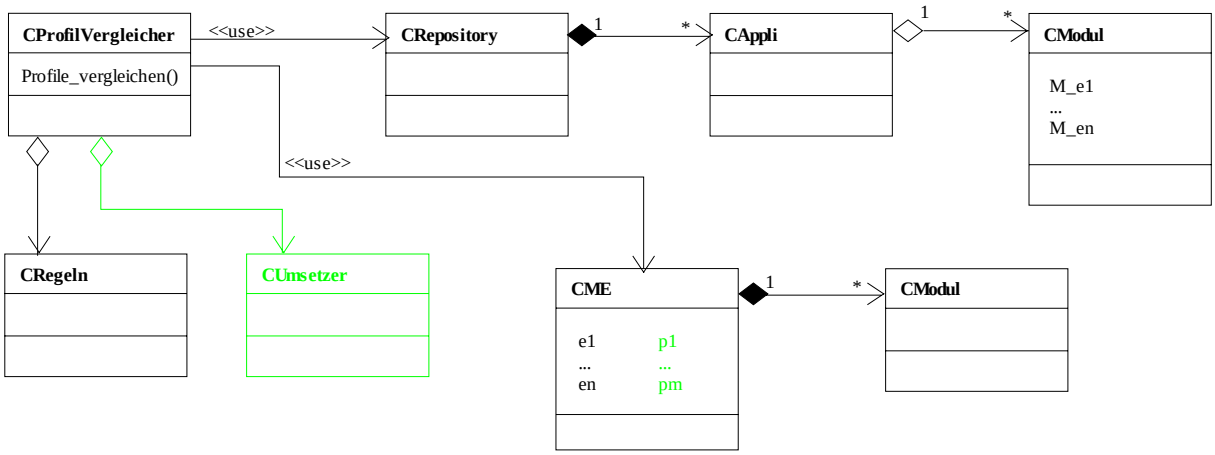

Abb. 1: Profiling Pattern

Das mobile Endgerät (Klasse CME) möchte sich automatisch mit den für ihn relevanten Softwaremodulen rekonfigurieren, die ein lokaler Server beherbergt (je nach Kontext, verschiedene Server mit unterschiedlichen Softwaremodulen: ortsabhängige Dienste). Das mobile Endgerät wird durch Profile charakterisiert, in denen seine Eigenschaften (e1 bis en oder p1 bis pm) stehen (Hardware, vorhandene Software, Benutzer Präferenzen). Im Repository des lokalen Servers (Klasse CRepository) liegen die konkreten Softwaremodule, dessen Eigenschaften (M_e1 bis M_en) auch durch ihre Profile beschrieben werden (für welchen Prozessortyp kompiliert, welches Display benötigt, ...). Applikationen (oder Dienste) bestehen aus mehreren Modulen. Eine Applikation (Klasse CAppli) wird auch durch ein Profil definiert, in dem die Eigenschaften der Applikation beschrieben werden (Version, ...) und die Liste der benötigten Module. Ein Applikationsmodul ist hier also ein abstraktes Konzept, welches konkret durch ein oder mehrere Softwaremodule realisiert wird (ein Applikationsmodul existiert z.B. als Softwaremodul für Prozessor xy und als Softwaremodul für Prozessor yz). Der Profile Matching Algorithmus (Klasse CProfilVergleicher) findet dann heraus, welche Softwaremodule vom lokalen Server dem mobilen Endgerät angeboten werden sollen, unter Berücksichtigung der schon vorhandenen Software auf dem mobilen Endgerät. Dazu benutzt er Regeln (Klasse CRegeln). Diesen Mechanismus kann man durch folgende Charakterisierungen verallgemeinern:

- Eigenschaften des Endgerätes: ME-Tupel = (eme1, ..., emen).
- Eigenschaften einer Applikation: App-Tupel = (eap1, ..., eapk, id1, ..., idq) wo eap1 bis eapn die Eigenschaften der Applikation sind, und id1 bis idq die Identifikatoren (z.B. Namen) der Applikationsmodule sind.
- Eigenschaften eines Softwaremoduls: Mo-Tupel = (id, emo2, ..., emom).

Der Profile Matching Algorithmus funktioniert folgendermaßen:

1. Welche Softwaremodule laufen überhaupt auf dem Endgerät? Regeln überprüfen, ob gewisse Anforderungen der Softwaremodule durch gewisse Eigenschaften des Endgerätes erfüllt werden (CUmsetzer übersetzt ggf. die Eigenschaften).
2. In dieser Untermenge von Softwaremodulen, wird jetzt für jede Applikation geprüft, ob auch noch alle nötigen Softwaremodule vorhanden sind, um die Applikation auf dem Endgerät zum Laufen zu bringen (nur lauffähige Applikationen werden behalten).
3. Von Modulen die zu einer Äquivalenzklasse gehören (selbe id) wird eines bevorzugt (z.B. ein Modul existiert in mehreren Versionen: die aktuellste wird bevorzugt). Applikationen können auch Prioritätsregeln haben. Hieraus entsteht die endgültige

Untermenge an Softwaremodulen, die dem Endgerät zum Runterladen angeboten werden.

3 Geplantes Szenario

3.1 Systembeschreibung

Bei dem geplanten Szenario sollen Sensorknoten von mobilen Robotern, die autonom durch das System fahren, rekonfiguriert werden. In unserem Labor kommen die von der Berkeley Universität entwickelten Mica2-Motes auf denen TinyOS[2] läuft und die von Fraunhofer AIS entwickelten Robertinos[3] zum Einsatz. Bei der Rekonfiguration ist folgendes zu beachten:

- Die Sensorknoten übernehmen die Rolle des mobilen Endgeräts, die Roboter die des lokalen Servers.
- Auf einem Knoten läuft nur eine Applikation, die komplett ersetzt werden muss.
- Die Applikation besteht aus mehreren Codefragmenten, die auf dem lokalen Server ausgesucht und auf diesem vor der Übertragung zu einer Applikation kompiliert werden (Content Adaption).

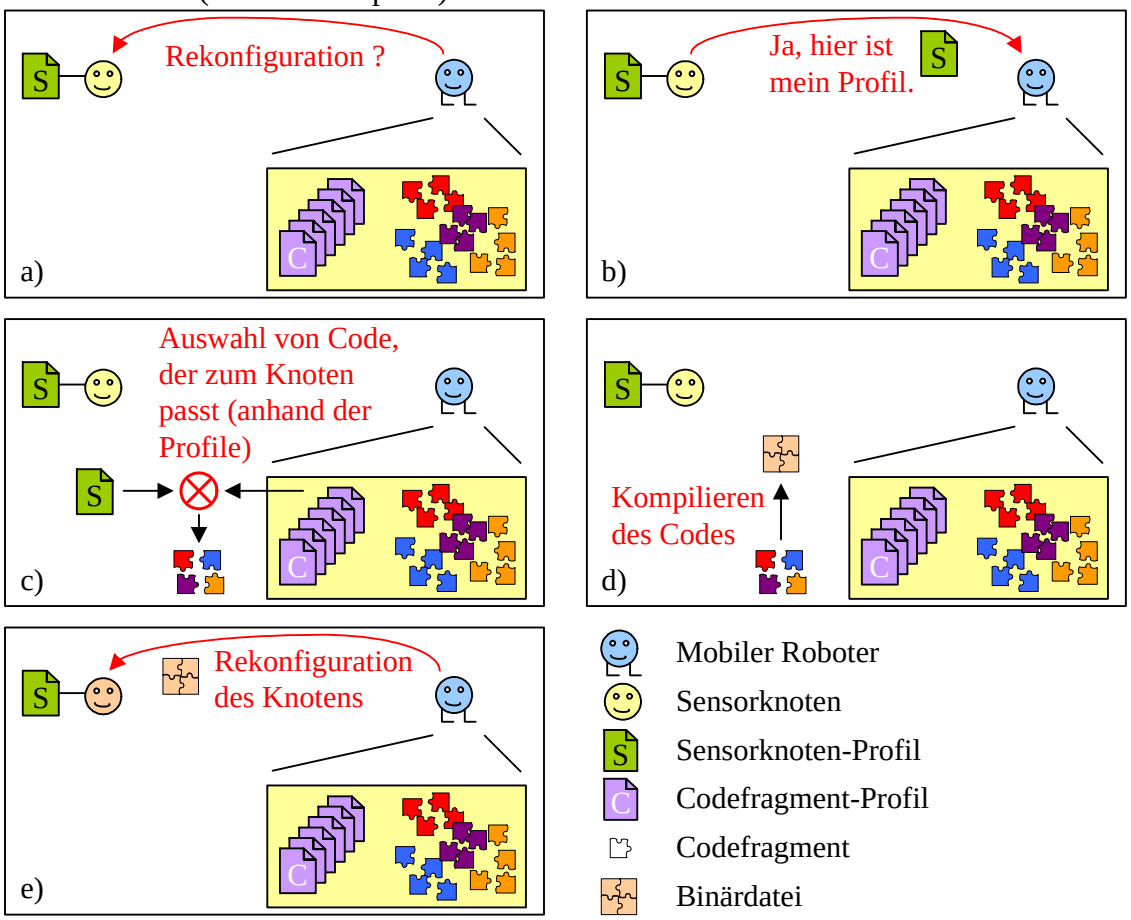

Abb. 2: Rekonfiguration eines Sensorknotens durch einen mobilen Roboter

3.2 Ablauf der Rekonfiguration

Ausgangssituation: Jeder Sensorknoten besitzt ein Profil, in dem seine Eigenschaften beschrieben sind (z.B.: Typ des Sensorknotens, Erweiterungshardware). Auf dem Roboter sind verschiedene Codefragmente abgelegt, aus denen eine neue Firmware für die Sensorknoten erstellt werden kann. Diese Codefragmente sind ebenfalls durch Profile

[2] Projekt-Homepage: http://www.tinyos.org
[3] Projekt-Homepage: http://www.openrobertino.org

beschrieben, in denen die Anforderungen an das Endgerät vermerkt sind (z.B.: Typ Mica2 + Sensor MTS510).

Abb. 2 zeigt den Ablauf der Rekonfiguration eines Sensorknotens mittels eines mobilen Roboters:

a) Der Roboter verkündet, dass eine Rekonfiguration nötig ist.
b) Will ein Sensorknoten die Rekonfiguration durchführen, sendet er sein Profil an den Roboter.
c) Der Roboter wählt anhand der Profile die Codefragmente aus, deren Anforderungen von den Eigenschaften des Sensorknotens erfüllt werden (*Profile Matching*).
d) Die ausgewählten Codefragmente werden kompiliert, so dass als Ergebnis eine für den Knoten angepasste Software entsteht (*Content Adaption*).
e) Die generierte Binärdatei wird auf den Sensorknoten übertragen, dieser startet neu, die Rekonfiguration ist abgeschlossen.

4 Erkenntnisse und weitere Vorgehensweise

Der hier vorgestellte Profiling Mechanismus wurde bereits prototypisch realisiert [4]. Die hierbei verwendeten mobilen Endgeräte waren im Vergleich zu den Sensorknoten (128kB Program Flash Memory, 38,4kBit/s Übertragungsrate) relativ mächtige PDAs mit WLAN-Anbindung. Es kamen ein symmetrisches Vokabular, externe Regeln und CC/PP (Composite Capabilities/Preference Profiles) zum Einsatz. Bei diesem Ansatz würde ein Sensorknoten für die komplette Beschreibung seiner Hardware / Konfiguration ca. 10% seines Speichers benötigen. Für das hier vorgestellte Szenario muss also eine ressourcenschonendere Umsetzung gefunden und die Content Adaption realisiert werden.

5 Literatur

[1] F. Dressler, B. Krüger, G. Fuchs, and R. German, "Self-Organization in Sensor Networks using Bio-Inspired Mechanisms," Proceedings of 18th ACM/GI/ITG International Conference on Architecture of Computing Systems - System Aspects in Organic and Pervasive Computing (ARCS'05): Workshop Self-Organization and Emergence, Innsbruck, Austria, March 2005, pp. 139-144.

[2] F. Dressler, "Sensor-Based Localization-Assistance for Mobile Nodes," Proceedings of 4. GI/ITG KuVS Fachgespräch Drahtlose Sensornetze, Zurich, Switzerland, March 2005, pp. 102-106.

[3] F. Dressler and G. Fuchs, "Energy-aware Operation and Task Allocation of Autonomous Robots," Proceedings of 5th IEEE International Workshop on Robot Motion and Control (IEEE RoMoCo'05), Dymaczewo, Poland, June 2005.

[4] G. Fuchs, "Mobile autonome Dienste und ihr Profiling," Master's Thesis (Diplomarbeit), Dept. of Computer Sciences, University of Erlangen-Nuremberg, 2004.

[5] T. Hakkarainen, A. Lattunen, and V. Savikko, "Flower - Framework for Local Wireless Services," in *ERCIM News*, vol. 50, pp. 51-52, July 2002.

[6] M. Hillenbrand, P. Müller, and K. Mihajloski, "A Software Deployment Service for Autonomous Computing Environments," Proceedings of International Conference on Intelligent Agents, Web Technology and Internet Commerce, July 2004.

[7] S. Truchat, "Interoperative Systems for Replenishment," Proceedings of Pervasive 2004, Doctoral Colloquium, Linz / Vienna, Austria, 2004.

[8] S. Truchat and A. Pflaum, "Reconfigurable comsumer direct logistics systems," Proceedings of 14. GI/ITG Fachtagung Kommunikation in Verteilten Systemen (KiVS), Kaiserslautern, Germany, March 2005.