SWARMS – Software Architecture for Radio-Based Mobile Self-Organizing Systems

Carsten Buschmann, Stefan Fischer

Institut für Betriebssysteme und Rechnerverbund
Technische Universität Braunschweig
Mühlenpfordtstr. 23, D-38106 Braunschweig
{buschmann|fischer}@ibr.cs.tu-bs.de

Jochen Koberstein, Norbert Luttenberger, Florian Reuter

Institut für Informatik und praktische Mathematik
Christan-Albrechts-Universität zu Kiel
Christian-Albrechts-Platz 4, D-24098 Kiel
{jko|nl|flr}@informatik.uni-kiel.de

1. Einleitung

Fortschritte hinsichtlich Miniaturisierung und Energieeffizienz von Hardware für Sensoren, Prozessoren und Speicher sowie Funkinterfaces haben mit den Sensornetzen eine neue Klasse von Netzwerken ermöglicht, die eine Fülle neuer Anwendungsfelder erschließen. So könnten beispielsweise über einem Waldbrandgebiet Tausende kleiner Temperaturfühler abgeworfen werden, die sich vernetzen und aufgrund der gemessenen Werte komplexere Informationen über den Brand wie etwa Position, Richtung und Geschwindigkeit von Feuerfronten liefern.

In dem beschriebenen Szenario finden sich Parallelen zu Schwärmen von z.B. Vögeln, Fischen oder Insekten, die mittels des Zusammenspiels von in ihren Fähigkeiten relativ beschränkten Individuen komplexe Aufgaben wie Nestbau oder Nahrungssuche bewältigen. In dem beschriebenen Netzwerk ergibt sich die Fähigkeit der einfachen Geräte, die Beantwortung komplexerer Fragestellungen zu unterstützen, ebenfalls durch Kooperation. Ihre große Zahl führt dabei zu neuen Herausforderungen hinsichtlich der Programmierung und Kooperation. Im SWARMS-Projekt wird untersucht, inwieweit die Programmierung von problemorientierten Anwendungen für gemeinsam operierende Schwärme von mobilen, funkvernetzten Systemen auf der Basis eines read/write Kooperationsparadigmas unterstützt werden kann [FL01]. In dieser Arbeit wird auf verschiedene Teilaspekte näher eingegangen: Zunächst wird der Begriff des Schwarmes näher erläutert. Im sich daran anschließenden Abschnitt wird das zugrunde liegende Datenmodell beschrieben. Der nächste Teil beschäftigt sich mit der geplanten Implementierung in einer Middleware. Der Beitrag wird durch die Betrachtung von Eigenschaften der avisierten Hardware abgerundet.

2. Schwärme

Das von uns betrachtete Kooperationsparadigma ist wie oben angedeutet durch das in der Tierwelt zu beobachtende Phänomen des Schwarms inspiriert. In Anlehnung daran wird der Begriff des Schwarmes hier wie folgt verstanden: Die *Einzelsysteme* verfügen über eine begrenzte Rechen- und Speicherleistung. Sie kommunizieren über ein infrastrukturloses Funknetzwerk, d.h. über ein geteiltes Medium. Dabei kann weder auf fehlerfreie noch auf ständige Konnektivität vertraut werden. Weiterhin muss davon ausgegangen werden, dass die meisten Einzelsysteme nicht über klassische Bedienelemente wie Tastatur oder Bildschirm verfügen; stattdessen sind sie über Sensoren oder Aktoren direkt in ihre Umwelt eingebettet. Hinsichtlich der Mobilität der Systeme lassen sich drei wesentliche Fälle unterscheiden. Im ersten Fall ist die weit überwiegende Anzahl der Geräte stationär (z.B. Sensorknoten). Im zweiten Fall sind die Geräte zwar mobil, die Applikation kann jedoch nicht über die Bewegung bestimmen (z.B. umher getragene PDAs). Der dritte Fall umfasst selbstbestimmte Bewegung, wofür autonome mobile Roboter ein Beispiel darstellen. Unabhängig von der

Mobilität muss damit gerechnet werden, dass eine optimale Platzierung der Systeme aufgrund von Anwendungsszenarien und Umweltgegebenheiten nicht möglich ist.

Die *Zusammensetzung* des Schwarmes ist variabel, die Veränderungen sind zumindest teilweise nicht im Voraus absehbar. Gründe für diese Veränderungen können menschliche Eingriffe sein, aber auch Mobilität, zeitweilige Kommunikationsabbrüche oder Gerätefehlfunktionen beispielsweise durch Batterieentleerung oder sogar Zerstörung. Somit sind die von einzelnen Systemen gelieferten Informationen weder persistent noch zuverlässig.

Der Schwarm verfolgt ein *definiertes gemeinsames Operationsziel*; dieses kann durchaus über die Zeit variieren. Es wird durch die auf alle Geräte einheitlich programmierte Anwendung vorgegeben und kann im Allgemeinen nur durch die Kooperation mehrerer Schwarmmitglieder erreicht werden. Dieses Ziel erfordert *situation awareness*: Die Geräte müssen sich des Kontextes bewusst sein, also z.B. ihrer Konfiguration, ihrer Umwelt oder ihrer eigenen Lokation. Letztere kann sowohl als absolute Position in Form von Koordinaten als auch kontextabhängig relativ (z. B. „am Ufer“) ausgedrückt werden. Dabei kann zur Ermittlung des Kontextes die Kooperation mit anderen Mitgliedern des Schwarmes nötig sein.

Aufgrund von Mobilität, sich wandelndem Kontext und funktionaler Beschränkung der Geräte ist eine feste Rollenzuweisung nicht sinnvoll. Daher haben die einzelnen Systeme *keine fest zugewiesenen Teilaufgaben*. Das Aushandeln der Aufgabenverteilung erfolgt selbstorganisiert und situationsabhängig. Beispielsweise könnte ein Gerät mit Sensoren und Festnetzanbindung je nach Bedarf die Funktionen Datenlieferant oder Festnetz-Gateway übernehmen.

3. Datenmodell

Die Lösung vieler Anwendungsprobleme setzt eine globale oder mindestens regionale Sicht auf die eigene Umgebung voraus. Nicht nur wegen der daraus resultierenden enormen Programmierkomplexität, sondern vor allem auch wegen des enormen Kommunikations- und damit verbundenen Energiebedarfs ist es nicht sinnvoll, diese Sicht durch den Austausch von direkt an einzelne Systeme gerichteten gerouteten Unicast-Nachrichten zu etablieren.

Daher fußt die Kooperation in SWARMS auf einem *Virtual Shared Information Space* (VSIS), in dem der Zustand sowohl der Umwelt als auch des Schwarmes selbst beschrieben ist. Es wird deshalb von einem „virtual“ shared information space gesprochen, weil es keinen Knoten im System gibt, der über alle Komponenten dieses Informationsraumes verfügt. Der VSIS setzt sich aus Informationen zusammen, die von den Schwarmmitgliedern akquiriert und verfügbar gemacht werden. Aus der Mobilität der Knoten, möglichen Geräteausfällen und der beschränkten Funkreichweite resultiert für jedes Mitglied eine unter Umständen inkonsistente Sicht auf einen Ausschnitt der Gesamtinformation.

Es gibt eine gewisse Ähnlichkeit zum Publisher/Subscriber-Modell [HG01], aber auch wesentliche Unterschiede: der VSIS kennt weder zentrale Instanzen zur Informationsverwaltung noch eine starre Rolleneinteilung in Informationsquellen und Senken.

Der VSIS wird aufgebaut aus semistrukturierten selbstbeschreibenden Informationskomponenten und durch ein XML-Schema in seinem Aufbau beschrieben. Diese Komponenten umfassen auch syntaktische und semantische Metainformationen zum Kontext der Informationsentstehung (räumlicher und zeitlicher Bezug, Verlässlichkeit) oder zu den

Daten selbst (Aggregationsgrad, Gültigkeitsbereich und -zeitraum). Diese helfen der Applikation, die Daten richtig zu nutzen.

4. Middleware

Um nun miteinander kommunizieren zu können, greifen Knoten mit Hilfe von Read-/Write-Operationen auf den VSIS zu. Die SWARMSware – eine im SWARMS-Projekt konzipierte Middleware – stellt eine Abstraktionsschicht dahingehend dar, dass der Programmierer eine Schwarm-Applikation als ein kohärentes Softwaresystem entwickeln kann, statt eine Menge unterschiedlicher Einzelsysteme zu programmieren.

Die SWARMSware verbirgt Selbstorganisation und Mobilität innerhalb eines Schwarms. Dazu gehören die Etablierung und Pflege von Ordnungen innerhalb des Schwarms (z.B. Regionen und Reihenfolgen). Der Aufwand zur Aufrechterhaltung solcher Ordnungen ist abhängig vom Grad der Mobilität der Schwarmmitglieder. Die SWARMSware stellt außerdem Mechanismen zur Etablierung eines gemeinsamen Zeitverständnisses zur Verfügung.

Das die VSIS-Struktur beschreibende XML-Schema legt fest, welche Struktur die Daten haben, die über die Zugriffsfunktionen der SWARMSware an die Anwendung übergeben werden können. Beim Zugriff der Applikation auf den VSIS erfolgt eine Transformation der Daten aus der XML-Darstellung in eine Repräsentation, die für die Applikation geeignet ist (Language Binding).

Beim Empfang von Informationen von anderen Knoten stellt die SWARMSware die Konformität der von anderen Knoten erhaltenen Daten zum XML-Schema sicher. Die Anwendung übergibt der SWARMSware Filter, die festlegen, welche dieser Informationen für sie von Interesse sind und in den VSIS-Ausschnitt aufgenommen werden sollen. Eine Herausforderung stellt die kompakte, die Anwendungsalgorithmen gut unterstützende und flexible Repräsentation der Daten dar. Die Applikation muss die Möglichkeit haben, sowohl die Anordnung als auch die Darstellung der Daten zu beeinflussen.

Eine weitere Aufgabe der Middleware ist die Unterstützung der *Kontextuierung und Bewertung von Informationskomponenten*. Metainformation wird z.B. durch die Applikation beim Schreibprozess oder durch die Middleware beim Erhalt von ähnlichen Daten (hinsichtlich Art, Zeit, Ort, etc.) erzeugt. Wenn ein Knoten z.B. weiß, dass mehrere direkte Nachbarn zu einem bestimmten Zeitpunkt eine Temperatur von 20°C messen, kann man aufgrund physikalischer Gesetze davon ausgehen, dass der von einem einzelnen Nachbarn übermittelte Messwert von 0°C nur bedingt vertrauenswürdig ist. Eine Möglichkeit zur Realisierung eines solchen Bewertungsmechanismus stellt das Voting mit Quoren dar, die von der Applikation spezifiziert werden. Dabei bleibt es der Anwendung überlassen, wie sie mit Informationen zur Verlässlichkeit von Daten umgeht. Modelliert werden solche Informationen entweder als XML-Attribute oder eigene XML-Dokumentenelemente.

Auf diesen Basismechanismen aufbauend unterstützt die Middleware zusätzlich die *Informationsaggregation*. Es werden Funktionen zur Transformation einer Menge von Informationselementen in eine kompaktere oder aussagekräftigere Darstellung zur Verfügung gestellt. Diese Transformationen sind anwendungsspezifisch und müssen nach von der Applikation vorgegebenen Regeln stattfinden.

Die Middleware soll auch die *Rollenverteilung* innerhalb des Schwarmes unterstützen. Die Aushandlung der Übernahme von verschiedenen Teilaufgaben durch einzelne Geräte erfolgt anhand des Kontextes und anhand der Fähigkeiten der Geräte. Der Kontext kann durch die Anwendung „gesetzt“ werden, oder nach im Vorfeld durch die Anwendung spezifizierten Kriterien von der Middleware ermittelt werden. Die Zuordnung von Aufgaben zu bestimmten Geräteeigenschaften und Kontextmerkmalen wird von der Anwendung nach ihren Bedürfnissen definiert.

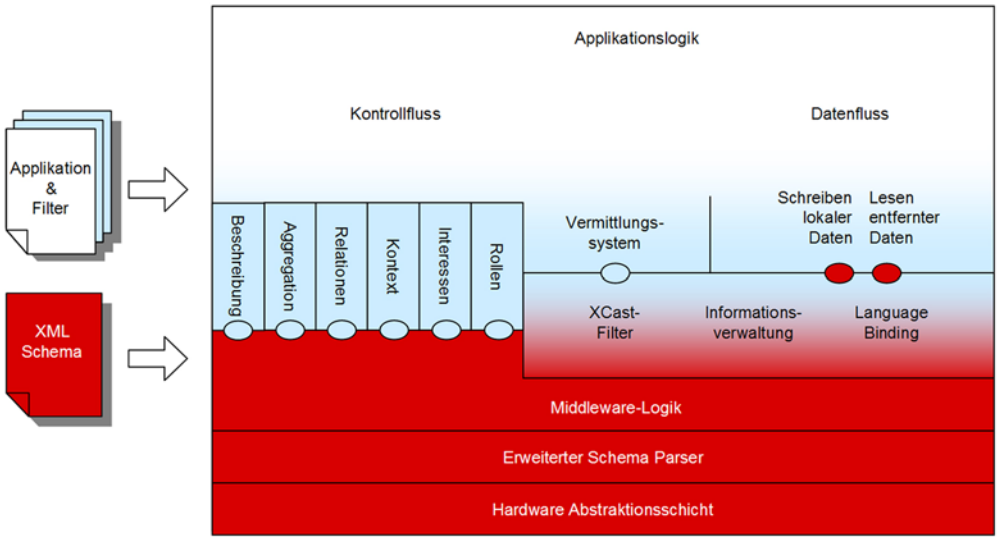

Abbildung 1: Architektur der SWARMSware mit der darauf aufsetzenden Anwendung

Im drahtlosen Netz eines Schwarms verbreiten die Knoten Informationen per adressenfreier Kommunikation (Broadcast). Knoten, die eine Information empfangen, leiten diese nur dann weiter, wenn sie für den Knoten selbst neu war bzw. mit eigener Information in geeigneter Art und Weise verschmolzen werden können. Ist die Menge der im Netz zu verteilenden Informationen endlich, lässt sich auf diese Weise die Anzahl der versendeten Pakete gegenüber simplem Fluten erheblich senken.

5. Hardware

Die von einigen Szenarien erforderte große Anzahl der benötigten Geräte sowie die große Verlustgefahr erfordern einen geringen Herstellungspreis. Darüber hinaus bedingen verschiedene Anwendungsszenarien eine starke Miniaturisierung [AAC+00], einige Forschungsgruppen legen ihren Überlegungen Geräte mit nur ein bis zwei Millimetern Kantenlänge zu Grunde [WLL+01]. Um die Batterielebensdauer der Geräte zu maximieren, ist ein geringer Stromverbrauch eine weitere Anforderung. Insgesamt muss davon ausgegangen werden, dass die in Sensornetzwerken eingesetzte Hardware eine sehr beschränkte Leistungsfähigkeit besitzt. So besitzen beispielsweise SensorMotes nur einen zwischen 8 und 128KB großen Befehlsspeicher sowie zwischen 512 Byte und 4 KByte Arbeitsspeicher [LC02]. Die Netzknoten kommunizieren über ein infrastrukturloses Funknetz; dabei handelt es sich um ein geteiltes Medium. Jeder Empfänger muss selbst entscheiden, ob eine ausgesendete Information für ihn interessant ist, es handelt sich um eine adressenlose Kommunikation. Somit findet auch kein Routing statt, es können höchstens Informationen in eine bestimmte Region geflutet werden. Da die Informationsübertragung mehr Energie benötigt als die Informationsverarbeitung [PG00], ist auf möglichst hohe Lokalität eingesetzter Verfahren zu achten.

6. Ausblick

Das SWARMS-Projekt befindet sich derzeit in der Anfangsphase. Neben einer groben Datenmodellierung wurden bisher Anforderungen und Ziele definiert, aber auch noch offene Forschungsfelder identifiziert. Die in dieser Arbeit beschriebenen Problemstellungen und Lösungsansätze sollen experimentell evaluiert und weiterentwickelt werden. Neben einer simulativen Erprobung soll auch mit einer heterogenen Handwarelandschaft und unterschiedlichen Kommunikationstechniken experimentiert werden. Dabei kommen PDAs (mit Bluetooth, WLAN, IrDA und RS-232-Schnittstelle), einfache mobile autonome Roboter (Kommunikation via IrDA) sowie Berkeley Mica Motes (mit einem 433MHz Funkinterface) zum Einsatz. Die verschiedenen Plattformen lassen sich miteinander kombinieren und bieten so eine breite, vielfältige Basis.

[AAC+00] Abelson, H., Allen, D., Coore, D., Hanson, C., Rauch, E., Sussmann, G.J., Weiss, R.: Amorphous Computing, Communications of the ACM, Vol. 43, No. 5, Mai 2000, 74-82

[FL01] Fischer, S., Luttenberger, N.: SWARMS – Software Architecture for Radio-Based Mobile Self-Organizing Systems, Projektantrag im Rahmen des DFG-Schwerpunktprogrammes „Basissoftware für selbstorganisierende Infrastrukturen für vernetzte mobile Systeme (SPP 1140), November 2001

[HG01] Huang, Y., Garcia-Molina, H. Publish/Subscribe in a Mobile Environment, International Workshop on Data Engineering for Wireless and Mobile Access, 2001, pp. 27-34

[LC02] Levis, P., Culler, D.: Maté: a Virtual Machine for Tiny Networked Sensors, ASPLOS, Dezember 2002.

[PG00] Pottie, G.J., Kaiser, W.J.: Wireless Integrated Network Sensors, Communications of the ACM, Vol. 43, No. 5, Mai 2000, 51-58

[WLL+01] Warneke, B., Last, M., Liebowitz, B., Pister, K.S.J.: Smart Dust: Communicating with a Cubic Millimeter Computer , IEEE Computer, 2001