Einsatz einer Echtzeit-Publish/Subscribe-Kommunikation für die Teleoperation mobiler Roboter

Timo Lindhorst, André Herms, Michael Schulze

Otto-von-Guericke-Universität Magdeburg
Institut für Verteilte Systeme
Timo.Lindhorst@st.ovgu.de, {aherms,mschulze}@ovgu.de

Zusammenfassung. In der Telerobotik werden mobile Roboter über ein Kommunikationsnetz durch einen Operator ferngesteuert. Um interaktiv zu operieren, müssen Daten in beide Richtungen in Echtzeit kommuniziert werden. Zusätzlich ist eine hohe Ausfallsicherheit des Systems zu gewährleisten. Diese Arbeit beschreibt die Umsetzung eines solchen Szenarios unter Verwendung einer Echtzeit-Publish/Subscribe-Kommunikation. Ein drahtloses Mesh-Netzwerk erfüllt durch redundante Verbindungen die Anforderungen an die Kommunikation. Eine Publish/Subscribe-Middleware ermöglicht eine inhaltsbasierte Kommunikation und erlaubt dadurch eine hohe Flexibilität der Teilnehmer. Durch die modulare Software-Architektur der Applikationen entsteht ein verteiltes System, in dem mobile Roboter durch netzwerktransparente Kommunikation gesteuert werden.

1 Einleitung

Mobile Roboter kommen bereits in zahlreichen Anwendungsgebieten zum Einsatz. Industriell werden sie z. B. in der Logistik als Transportsystem verwendet, aber auch im militärischen Bereich oder im Katastropheneinsatz eröffnen sich diverse Anwendungsszenarien.

Die Mobilität der Roboter erfordert zum Datenaustausch zwingend eine drahtlose Kommunikation. Je nach Anwendung werden unterschiedliche Anforderungen an die Datenübertragung gestellt. Zum einen ist eine Fernsteuerung denkbar, so dass der Roboter über ein Netz Steuersignale empfängt und entsprechend umsetzt. Zum anderen sendet der Roboter seinerseits Informationen an andere Teilnehmer im Netz. Denkbar ist beispielsweise die Übertragung des Videobildes einer auf dem Roboter angebrachten Kamera, um so eine Fernsteuerung über Sichtgrenzen hinweg zu ermöglichen. Darüber hinaus ist auch die Übermittlung verschiedener Sensordaten sinnvoll.

Die genannten Beispielanwendungen stellen Echtzeitanforderungen an das Kommunikationssystem. Eine unterbrechungsfreie Datenübertragung muss gewährleistet sein. Dabei ist zu beachten, dass der Roboter auf Grund seiner Mobilität während der Übertragung Empfangsbereiche verlässt und neue erschließt.

Das Kommunikationsnetz muss eine hohe Ausfallsicherheit aufweisen. Um Flexibilität zu gewährleisten, sollte der Aufwand zur Integration weiterer Teilnehmer im Netz möglichst gering sein.

Das in dieser Arbeit beschriebene System wird den genannten Anforderungen gerecht, indem es einerseits ein drahtloses Mesh-Netzwerk für die Datenübertragung verwendet und andererseits eine inhaltsbasierte Kommunikation nach dem Publish/Subscribe-Verfahren einsetzt. Neben den Echtzeitanforderungen, die eine Steuerung bedingt, steht eine hohe Flexibilität des Systems im Vordergrund.

Im folgenden Abschnitt 2 wird zunächst das Szenario beschrieben und damit die Anforderungen an das System definiert. Abschnitt 3 beschreibt die Softwarearchitektur im Detail. Abschließend wird in Abschnitt 4 ein Fazit gezogen und herausgestellt, inwieweit die Umsetzung des Systems den Anforderungen gerecht wird.

2 Szenario

Ein mobiler Roboter wird durch Teleoperation über größere Entfernungen hinweg gesteuert. Dabei sind der Operator und der Roboter über ein Kommunikationsnetz miteinander verbunden. Über das Netz werden zum einen Steuersignale vom Operator zum Roboter gesendet, zum anderen erfordert eine interaktive Steuerung, dass dem Operator Informationen vom Roboter übermittelt werden, z. B. ein Kamerabild oder Sensorwerte.

Das Szenario stellt verschiedene Anforderungen an die Kommunikation. Die Mobilität des Roboters erfordert eine drahtlose Datenübertragung. Um eine durchgängige Kommunikation zwischen Operator und Roboter zu ermöglichen, muss das gesamte Operationsgebiet durch das Netz abgedeckt sein. Des Weiteren stellt die interaktive Steuerung des Roboters Echtzeitanforderungen an die Kommunikation. Da drahtlose Kommunikation störanfällig ist, beziehen wir uns auf weiche Echtzeit: Die Daten werden im Allgemeinen mit geringen Latenzen übertragen, dennoch muss das System auch mit Verzögerungen umgehen können. Der Roboter muss beispielsweise beim Ausbleiben von Steuerkommandos in einen sichern Zustand wechseln.

Um eine hohe Ausfallsicherheit zu gewährleisten, bedarf es redundanter Kommunikationspfade zwischen Operator und Roboter. Da auch der Operator oder der Roboter ausfallen kann, besteht als weiterer Anspruch eine hohe Flexibilität und einfache Konfigurierbarkeit des Systems. Ein neuer Roboter sollte ohne zusätzlichen Aufwand von einem beliebigen Operator im Netz gesteuert werden können.

Im Folgenden werden der Roboter und die Umgebung vorgestellt, in der das beschriebene Szenario umgesetzt ist.

2.1 Roboter

Bei dem zu steuernden mobilen Roboter (s. Abbildung 1) handelt es sich um einen VolksBot [1], ein modulares Prototypen Robotersystem des Fraunhofer

Abb. 1. VolksBot

Instituts für Intelligente Analyse- und Informationssysteme (IAIS). Als zentrale Steuereinheit dient ein auf dem Roboter platziertes Notebook, das über USB und verschiedene Konverter mit den verfügbaren Komponenten (Sensoren und Aktoren) verbunden ist. Die Erweiterung des Roboters um weitere Komponenten ist daher problemlos möglich.

Das Fahrwerk besteht aus einem Differentialantrieb und zwei frei beweglichen Stützrädern. Die Steuerung übernimmt ein Motor-Controller. Auf diese Weise ist die Regelung des Antriebs isoliert von anderen Prozessen.

Mit acht gleichmäßig um den Rumpf des Roboters angeordneten Ultraschallsensoren kann die Umgebung nach Hindernissen abgesucht werden. Durch einen Liniensensor in der Bodenplatte des Roboters ist es möglich, eine kontrastreiche Spur auf dem Boden zu erkennen. Des Weiteren sind zwei Kameras und ein Greifarm vorhanden.

Die Kommunikation des Roboters mit seiner Umwelt ist über die WLAN-Schnittstelle des Notebooks möglich. Sie unterstützt die gängigen Standards nach IEEE 802.11 und ermöglicht somit die Einbindung des Roboters in ein drahtloses Netzwerk.

2.2 Umgebung

Die Steuerung des Roboters erfolgt innerhalb einer Etage unseres Gebäudes (s. Abbildung 3). Dies bedingt, dass die gesamte Etage durch ein drahtloses Netz abgedeckt ist. In verschiedenen Räumen sind Linux-Workstations und Access Points (APs) verteilt, die dieses gemeinsame Netz bilden. Das auf den APs laufende Linux-System kann in der Funktionalität angepasst werden.

Jede Workstation ist als Operator einsetzbar. Außerdem müssen Roboter im Netz bei ihrer Aktivierung ohne vorherige Konfiguration sofort von einem Operator steuerbar sein. Informationen der Roboter (Kamerabilder oder Sensorwerte) können neben dem Operator auch von beliebigen weiteren Rechnern im Netz empfangen werden. Die überwachenden Stationen benötigen keine Kenntnis über derzeit aktive Roboter. Aktuelle sensorische Informationen der Roboter stehen

zur Verfügung, ohne dass eine direkte Anfrage an die einzelnen Roboter gerichtet werden muss.

Darüber hinaus soll eine synchrone Steuerung mehrerer Roboter möglich sein. Das bedeutet, dass Steuersignale vom Operator an alle im Netz verfügbaren Roboter gesendet werden, ohne dass der Operator Kenntnisse darüber hat, welche Roboter derzeit im Netz vorhanden sind.

3 Software-Architektur

Die Softwarearchitektur des Systems ist in Abbildung 2 schematisch dargestellt. Die einzelnen Schichten werden im Folgenden erläutert.

Wie bereits in Abschnitt 2 beschrieben, stellt die Steuerung eines Roboters über ein Netz Echtzeitanforderungen an die Datenübertragung. Darüber hinaus erfordert die Mobilität eine erhöhte Flexibilität des Netzes. In den Abschnitten 3.2 und 3.3 wird erläutert, wie diese Anforderungen durch drahtlose Mesh-Netzwerke und inhaltsbasierte Adressierung erfüllt werden. Die für diesen Zweck verwendeten Systeme, AWDS und COSMIC, werden vorgestellt. Abschnitt 3.4 beschreibt die modulare Applikationsschicht, die auf GEA, einer Plugin-Architektur mit zentraler Ereignisverarbeitung, aufsetzt und COSMIC als Kommunikationssystem nutzt. Schließlich wird in Abschnitt 3.5 gezeigt, wie durch Komposition verschiedener Module unterschiedliche Applikationen entstehen.

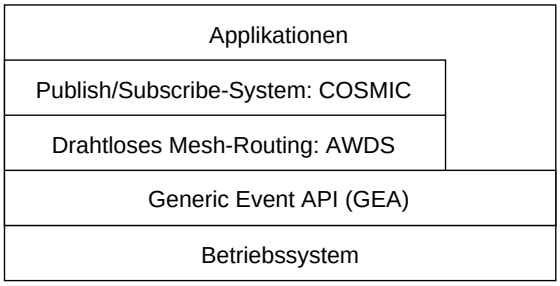

Abb. 2. Software-Architektur der Robotersteuerung

3.1 Generic Event API

Der modulare Aufbau der Roboter-Hardware legt einen ebenfalls modularen Aufbau der Software nahe (s. a. Abschnitt 3.4). GEA (**G**eneric **E**vent **A**PI, [2]) bietet eine zentrale Ereignisverarbeitung und damit eine Sequentialisierung der nebenläufigen Vorgänge. Als Plugin-Architektur ermöglicht GEA, einzelne Module zur Laufzeit nachzuladen und unterstützt damit die Modularität der Applikation. Darüber hinaus basiert auch das im folgenden Abschnitt beschriebene Mesh-Routing Protokoll AWDS auf GEA.

3.2 Drahtloses Mesh-Netzwerk

Das in Abschnitt 2 beschriebene Szenario stellt diverse Anforderungen an die drahtlose Kommunikation. Zunächst ist ein großes Gebiet abzudecken. Dabei soll das Netz möglichst flexibel aufgebaut sein. Die Steuerung eines Roboters erfordert weiterhin Echtzeitfähigkeit und eine hohe Ausfallsicherheit.

Diesen Anforderungen werden drahtlose Mesh-Netzwerke gerecht. Im Gegensatz zu herkömmlichen Infrastruktur-Netzen werden gleichzeitig Verbindungen zu allen erreichbaren Stationen aufgebaut. Auf diese Weise entfallen zeitaufwendige Roaming-Prozesse beim Durchqueren verschiedener Funkzellen. Durch die redundanten Verbindungen innerhalb des Netzes wird eine erhöhte Echtzeitfähigkeit und Ausfallsicherheit erreicht. Darüber hinaus ermöglicht die Selbstorganisation drahtloser Mesh-Netze eine einfache, kostengünstige und flexible Erweiterung der Netzabdeckung.

Für unsere Arbeit verwenden wir AWDS (**A**d-Hoc **W**ireless **D**istribution **S**ervice, [3]), um ein drahtloses Mesh-Netz aufzubauen. Bei AWDS handelt es sich um eine Open-Source Implementierung eines proaktiven Link-State-Routings auf der Sicherungsschicht.

Um die in Abschnitt 2.2 beschriebene Abdeckung der Umgebung zu erreichen, kann AWDS sowohl auf den Workstations als auch auf den APs ausgeführt werden. Mit der Anzahl der Stationen steigt die Zahl der möglichen Verbindungen und damit die Ausfallsicherheit.

Schließlich wird auch der Roboter über AWDS in das Netz integriert. Die sich daraus ergebende Topologie ist in Abbildung 3 dargestellt. Bei den Knoten `d41`x handelt es sich um mobile Roboter, die anderen Knoten sind Workstations oder APs. AWDS unterscheidet die verschiedenen Stationen dabei nicht in ihrer Funktionalität. Alle Stationen können Daten senden, empfangen und weiterleiten.

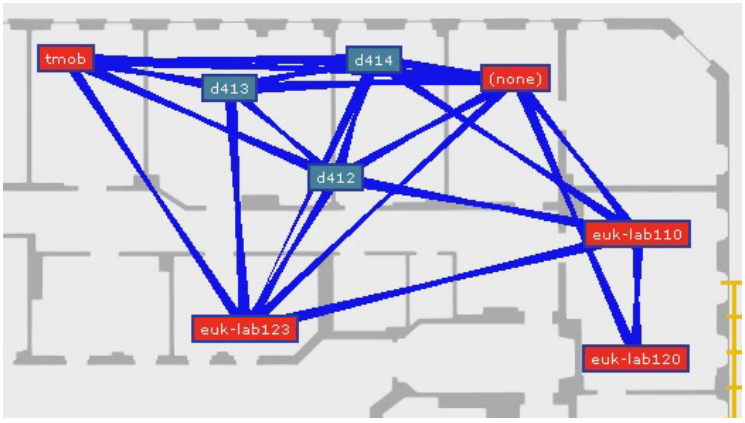

Abb. 3. Topologie des Mesh-Netzes

AWDS ermöglicht eine drahtlose Kommunikation, die mobile Teilnehmer erlaubt und durch redundante Verbindungen sicherstellt, dass auch bei ausgefallenen Stationen oder beim Verlassen eines Empfangsbereichs die Datenübertragung nicht unterbrochen wird. Dies ermöglicht die Echtzeit-Steuerung eines mobilen Roboters. Durch Selbstorganisation wird ein hohes Maß an Dynamik innerhalb des Netzwerkes erreicht.

3.3 Inhaltsbasierte Kommunikation

Unser Szenario fordert eine hohe Flexibilität, damit sowohl Roboter als auch der Operator ohne zusätzlichen Konfigurationsaufwand ausgetauscht werden können. Müssen beim Datenaustausch die Kommunikationspartner direkt adressiert werden, wird die Flexibilität eingeschränkt. Soll beispielsweise eine Überwachungseinheit die Sensoren der aktiven Roboter überwachen, so muss zunächst bekannt sein, welche Roboter verfügbar sind. Diese müssen adressiert werden, um die Werte abfragen zu können. Wird ein weiterer Roboter aktiviert, so müsste die Überwachungseinheit darüber informiert werden, um auch diesen Roboter zu berücksichtigen. Der Roboter selbst hat aber keine Kenntnis über die Überwachungseinheit und kann sie demnach nicht benachrichtigen.

Im Allgemeinen ist aber der Inhalt der Daten von Bedeutung und nicht der Kommunikationspartner, von dem diese Daten bezogen werden. Bei einer inhaltsbasierten Kommunikation nach dem Publish/Subscribe-Verfahren werden daher Inhalte statt Stationen adressiert. Die Kommunikationsteilnehmer sind durch eine Middleware entkoppelt, welche sicher stellt, dass Daten eines bestimmten Typs an die entsprechenden Interessenten weitergeleitet werden.

Eine Kommunikation nach dem o. g. Beispiel läuft demnach wie folgt ab: Die Überwachungseinheit meldet sich bei der Middleware als Subscriber für Daten vom Typ „Distanz" an. Wird von einem Distanz-Sensor am Roboter eine Messung abgeschlossen, wird dieser Wert mit dem Typ „Distanz" über die Middleware publiziert. Da die Middleware die Überwachungseinheit als Subscriber registriert hat, werden die Daten im Netz übertragen und eine Nachricht entsprechend des gewünschten Typs an die Überwachungseinheit ausgeliefert.

Für die Umsetzung unseres Szenarios verwenden wir COSMIC (**CO**operating **SM**art **DE**vices, [4,5]), eine Publish/Subscribe-Middleware, die verschiedene Kommunikationsnetze, wie z. B. CAN und TCP/IP unterstützt. COSMIC-Ereignisse sind durch ein Subjekt eindeutig mit einem definierten Inhalt verknüpft. Diese Zuordnung ist über das gesamte Netz eindeutig und bildet somit einen globalen Adressraum. Will eine Anwendung Ereignisse publizieren oder empfangen, so muss sie einen Ereigniskanal anfordern welcher mit einem Subjekt verknüpft ist. COSMIC reserviert dadurch die nötigen Ressourcen im Netzwerk.

COSMIC ermöglicht eine dynamische, inhaltsbasierte Kommunikation. Teilnehmer können im Netz auftauchen und verschwinden und sind dabei nicht auf die Kenntnis von Adressen angewiesen, um zu kommunizieren.

Durch die Erweiterung von COSMIC um die Unterstützung von AWDS als zusätzliches Kommunikationsmedium [6,7], vereint man die Vorteile beider Sys-

teme und erreicht so eine drahtlose, flexible Infrastruktur zur Kommunikation zwischen stationären und mobilen Teilnehmern.

3.4 Applikationsschicht

Auf Grund des modularen Aufbaus des Roboters ist auch die Software modular aufgebaut, um die Erweiterung um zusätzliche Komponenten zu ermöglichen. Für jeden Sonsortyp und jeden Aktor wird ein Modul erstellt. Darüber hinaus gibt es verschiedene Module, die eine Regelung mit Hilfe der Sensor/Aktor-Module umsetzen. Weitere Module ermöglichen die Fernsteuerung des Roboters und die Verbreitung verschiedener Sensorwerte über ein Netz.

Durch GEA (vgl. Abschnitt 3.1) ist es möglich, verschiedene Module zu laden. Abhängig von der Wahl der Module ergibt sich eine bestimmte Funktionalität der Applikation. Das kann zum einen eine Applikation sein, die auf dem Roboter die Steuerung übernimmt, zum anderen wird auch die Operator-Applikation zur Fernsteuerung des Roboters durch eine entsprechende Modul-Komposition gebildet. Die Module und ihre Abhängigkeiten untereinander werden in Abbildung 4 dargestellt und im Folgenden näher erläutert.

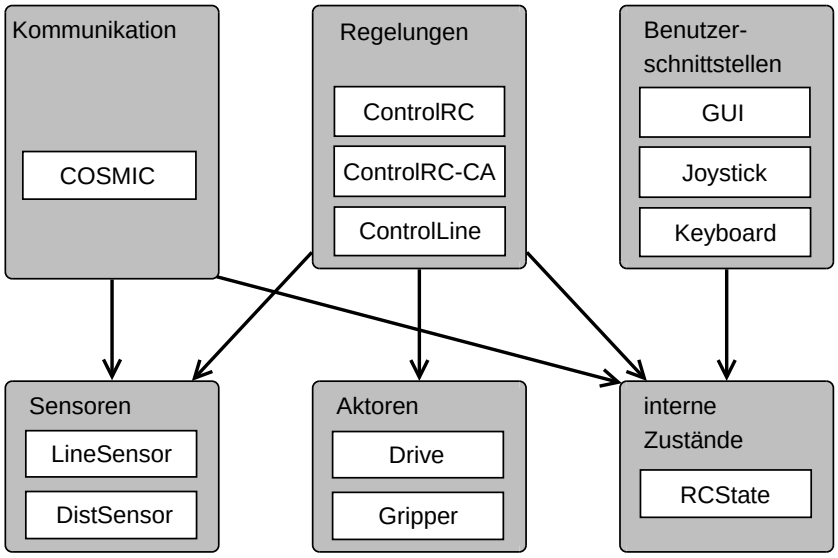

Abb. 4. Applikationsmodule und ihre Abhängigkeiten untereinander

Sensoren. Für jeden Sensortyp ist ein eigenes Modul vorhanden. Es können beliebige *Sensor*-Module parallel geladen werden. Die Werte der Sensoren können von anderen Modulen über eine definierte Schnittstelle abgefragt werden. Ist

COSMIC als Kommunikationsmodul geladen, werden die Werte darüber hinaus im Netz publiziert.

Das *LineSensor*-Modul überwacht den Liniensensor. Das *DistSensor*-Modul löst periodisch Messvorgänge bei den Distanzsensoren aus und fragt die Messwerte ab. Dabei wird berücksichtigt, dass sich die einzelnen Sensoren bei der Messung nicht beeinflussen.

Aktoren. Die verschiedenen Aktoren des Roboters werden über unterschiedliche Module mit definierten Schnittstellen steuerbar. *Aktor*-Module beeinflussen sich untereinander nicht und können demnach parallel geladen werden. Das *Drive*-Modul steuert den Antrieb, das *Gripper*-Modul den Greifarm.

Interne Zustände. Interne Zustände werden eingeführt, um beispielsweise Kommandos einer Fernsteuerung nicht direkt auf den Aktoren oder der Steuerung umzusetzen, sondern lediglich einen bestimmten Soll-Zustand für die Regelung zur Verfügung zu stellen.

Das *RCState*-Modul verwaltet beispielsweise die durch die Fernsteuerung vorgegebene Soll-Geschwindigkeit der einzelnen Räder. Das bedeutet, eine Fernsteuerung ändert zunächst nur diesen Zustand. Der entsprechende Regler versucht die Soll-Geschwindigkeit umzusetzen.

Im Zusammenspiel mit COSMIC können Zustände, wie die Soll-Geschwindigkeit im *RCState*-Modul, entweder als Publisher oder als Subscriber betrachtet werden. Bei einer Operator-Applikation ändert beispielsweise ein Joystick den Zustand, der daraufhin durch COSMIC publiziert wird. Auf dem Roboter bildet der Zustand einen Subscriber und wird entsprechend über das *COSMIC*-Modul aktualisiert. Die Kommunikation zwischen den Modulen im verteilten System erfolgt durch COSMIC.

Regelungen. Regelungsmodule steuern den Roboter. Dabei kann jeweils nur ein entsprechendes Modul geladen werden. Je nach Modul entstehen Abhängigkeiten zu verschiedenen Aktoren, Sensoren und internen Zuständen.

Das *ControlRC*-Modul setzt beispielsweise eine Fernsteuerung um. Demnach bedingt es das *RCState*-Modul und den Antrieb. Das *ControlRC-CA*-Modul entspricht dem *ControlRC*-Modul, vermeidet darüber hinaus aber Kollisionen, indem es Werte des Distanz-Sensors berücksichtigt. Eine Spurverfolgung kann über ein *ControlLine*-Modul umgesetzt werden, welches lediglich den Antrieb und den Liniensensor benötigt.

Benutzerschnittstellen. Zur Teleoperation des Roboters stehen verschiedene Benutzerschnittstellen zur Verfügung. Das *Joystick*- und das *Keyboard*-Modul ermöglichen eine Steuerung über einen Joystick oder die Tastatur. Bei einer Fernsteuerung werden die Kommandos mittels COSMIC über einen internen Zustand publiziert. Zur Visualisierung der Sensorwerte existiert ein *GUI*-Modul, welches über COSMIC als Subscriber an die entsprechenden Werte gelangt (s. Abbildung 5).

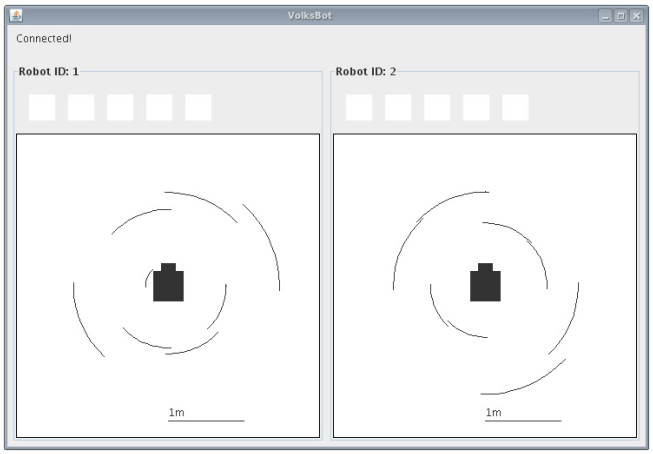

Abb. 5. Visualisierung der Distanz-Werte

Kommunikation. Das *COSMIC*-Modul realisiert die in Abschnitt 3.3 beschriebene Publish/Subscribe-Kommunikation. Ist das Modul geladen, so werden die verfügbaren Sensorwerte publiziert und die Subscriber erhalten die angeforderten Daten. Die Middleware verbindet auf diese Weise ein verteiltes System, indem sie das Kommunikationsnetz vor den Modulen verbirgt.

3.5 Beispielapplikationen

Durch verschiedene Kompositionen der verfügbaren Module können unterschiedliche Applikationen gebildet werden. Durch die Publish/Subscribe-Middleware können die Module netzwerktransparent kommunizieren. Ein Steuermodul (z. B. für den Joystick) sowohl direkt auf dem Roboter als auch auf einem Operator ausgeführt werden, wobei die Kommandos durch das *COSMIC*-Modul zum Roboter übertragen werden.

Soll der Roboter über einen direkt angeschlossenen Joystick gesteuert werden, so wird in der Minimalkonfiguration nur das *Joystick*-, das *Drive*-, das *RCState*-, und das *ControlRC*-Modul benötigt.

Soll der Roboter per Teleoperation über das Mesh-Netz gesteuert werden, entsteht die entsprechende Applikation, indem neben dem *Drive*-, *ControlRC*- und dem *RCState*-Modul das *COSMIC*-Modul geladen wird. Das *RCState*-Modul ist in diesem Fall ein Subscriber und erhält über COSMIC die Steuerkommandos. Die Operator-Applikation, die auf einer Workstation läuft, um den Roboter zu steuern, besteht aus dem *Joystick*-, dem *RCState*- und dem *COSMIC*-Modul. Dabei werden Zustandsänderungen des *RCState*-Moduls über COSMIC publiziert. Sollen auf einer Workstation lediglich die Sensorwerte überwacht werden, so genügt das Laden des *GUI*- und des *COSMIC*-Moduls.

4 Fazit und Ausblick

In dieser Arbeit präsentieren wir den Einsatz einer Echtzeit-Publish/Subscribe-Kommunikation in einem real umgesetzten Teleoperations-Szenario. Ein Roboter wird über ein drahtloses Netz von einem Operator ferngesteuert und stellt dabei anderen Stationen Informationen zur Verfügung. Die Verwendung eines drahtlosen Mesh-Netzes bietet durch redundante Verbindungen eine hohe Ausfallsicherheit und ermöglicht eine Echtzeit-Steuerung des Roboters. Die Einsatzumgebung ist durch die Selbstorganisation des Netzes einfach erweiterbar und flexibel.

Die eingesetzte Publish/Subscribe-Kommunikation unterstützt den dynamischen Charakter des Mesh-Netzes und ermöglicht durch inhaltsbasierte Kommunikation eine einfache Austauschbarkeit von Operator und Robotern.

Die modulare Software-Architektur ermöglicht eine flexible Anpassung der Applikationen an die jeweiligen Anforderungen. Durch den Einsatz der Publish/Subscribe-Middleware zur netzwerktransparenten Kommunikation zwischen den Modulen entsteht ein verteiltes System zur Teleoperation mobiler Roboter.

Gegenwärtig bestehen noch offene Fragestellungen bei der rechtzeitigen Erkennung von Verbindungsabbrüchen. Diese lassen sich nicht optimal in der Routingschicht feststellen. Daher konzentrieren sich die weiteren Arbeiten auf dieses Problem, welches durch einen schichtübergreifenden Ansatz gelöst werden soll. Dabei lösen Ereignisse in der Sicherungsschicht Reaktionen in den höheren Schichten des Protokollstapels aus. Hierdurch ist es möglich rechtzeitig alternative Routen festzulegen.

Literaturverzeichnis

1. VolksBot Homepage, 2008. online, http://www.volksbot.de/
2. A. Herms und D. Mahrenholz: *Unified Development and Deployment of Network Protocols.* Proceedings of Meshnets, Budapest, Hungary, 2005
3. AWDS Projekt Homepage, 2008. online, http://awds.berlios.de
4. J. Kaiser und C. Brudna: *A Publisher/Subscriber Architecture Supporting Interoperability of the CAN-Bus and the Internet.* In 2002 IEEE International Workshop on Factory Communication Systems, Väesteras, Schweden, 2002.
5. M. Schulze und S. Zug: *Using COSMIC – A real world case study combining virtual and real sensors.* In Proceedings of the 5th Minema Workshop on Middleware for Network Eccentric and Mobile Applications, Magdeburg, Germany, 2007.
6. A. Herms und M. Schulze: *Publish/Subscribe Middleware für Selbstorganisierende Drahtlose Multi-Hop-Netzwerke.* Workshop über Selbstorganisierende, Adaptive, Kontextsensitive verteilte Systeme, Wiesbaden, Germany, 2008
7. A. Herms, M. Schulze, J. Kaiser und E. Nett: *Exploiting Publish/Subscribe Communication in Wireless Mesh Networks for Industrial Scenarios.* In Proceedings of the 13th. IEEE Internatinal Conference on Emerging Technologies and Factory Automation (ETFA), Hamburg, Germany, September 2008