Informationsaustausch zwischen mobilen Teilnehmern – Eine Lösung unter Verwendung von Ad-Hoc Kommunikation und verteilter Datenspeicherung

Jens Wittrowski, Siemens AG

C-LAB Report

Vol. 9 (2010) No. 03

Cooperative Computing & Communication Laboratory

ISSN 1619-7879

C-LAB ist eine Kooperation
der Universität Paderborn und der Siemens AG
www.c-lab.de
info@c-lab.de

C-LAB Report

Herausgegeben von
Published by

Dr. Wolfgang Kern, Siemens AG
Prof. Dr. Franz-Josef Rammig, Universität Paderborn

Das C-LAB - Cooperative Computing & Communication Laboratory - leistet Forschungs- und Entwicklungsarbeiten und gewährleistet deren Transfer an den Markt. Es wurde 1985 von den Partnern Nixdorf Computer AG (nun Siemens AG) und der Universität Paderborn im Einvernehmen mit dem Land Nordrhein-Westfalen gegründet.
Die Vision, die dem C-LAB zugrunde liegt, geht davon aus, dass die gewaltigen Herausforderungen beim Übergang in die kommende Informationsgesellschaft nur durch globale Kooperation und in tiefer Verzahnung von Theorie und Praxis gelöst werden können. Im C-LAB arbeiten deshalb Mitarbeiter von Hochschule und Industrie unter einem Dach in einer gemeinsamen Organisation an gemeinsamen Projekten mit internationalen Partnern eng zusammen.

C-LAB - the Cooperative Computing & Cooperation Laboratory - works in the area of research and development and safeguards its transfer into the market. It was founded in 1985 by Nixdorf Computer AG (now Siemens AG) and the University of Paderborn under the auspices of the State of North-Rhine Westphalia.
C-LAB's vision is based on the fundamental premise that the gargantuan challenges thrown up by the transition to a future information society can only be met through global cooperation and deep interworking of theory and practice. This is why, under one roof, staff from the university and from industry cooperate closely on joint projects within a common research and development organization together with international partners. In doing so, C-LAB concentrates on those innovative subject areas in which cooperation is expected to bear particular fruit for the partners and their general well-being.

ISSN 1619-7879

C-LAB
Fürstenallee 11
33102 Paderborn
fon: +49 5251 60 60 60
fax: +49 5251 60 60 66
email: info@c-lab.de
Internet: www.c-lab.de

Inhaltsverzeichnis

Informationsaustausch zwischen mobilen Teilnehmern – Eine Lösung unter Verwendung von Ad-Hoc Kommunikation und verteilter Datenspeicherung

1 Einleitung

Im Rahmen des mit Mitteln des Bundesministeriums für Wirtschaft und Technologie [BMWI] innerhalb des Förderschwerpunkts SimoBIT [SIMOBIT] geförderten Projektes „Robot2Business“ [R2B] werden unter anderem Lösungen für die Integration von mobilen Teilnehmern in die Geschäftsprozesse von Unternehmen erarbeitet.

Als Industriepartner für Anwendungsszenarien sind der IT-Dienstleister „*CADsys Vertriebs- und Entwicklungs GmbH“* und der Landmaschinenhersteller „*CLAAS Selbstfahrende Erntemaschinen GmbH“* im Projekt vertreten, wobei letzterer die Konsortialführerschaft im Projekt koordiniert. Im Anwendungsszenario von CADsys sind die mobilen Teilnehmer die Notebooks der Mitarbeiter, während es sich bei CLAAS um die Landmaschinen handelt (z.B. Feldhäcksler, Traktor, etc).

Dieser Report befasst sich mit zwei Basisdiensten, welche zur Erreichung des Gesamtziels (Integration der mobilen Teilnehmer in Geschäftsprozesse) erforderlich sind und im Wesentlichen aus Anwendungsszenarien der Firma CLAAS hervorgegangen sind.

Für die Modellierung und Umsetzung der Geschäftsprozesse wird eine IT-Umgebung auf Basis einer serviceorientierten Architektur [SOA] aufgebaut, bei der die Systeme via Webservices kommunizieren. Die mobilen Teilnehmer müssen daher in diese SOA integriert werden und die Kommunikation über Webservices unterstützen.

Eine Webservices-Kommunikation setzt voraus, dass die vollständige URL der aufzurufenden Webservices bekannt ist. Eine URL besteht aus einer Adresse (IP Adresse), einem Port und einem Pfad für den Webservice und der zu verwendenden Operation. In statischen IT-Umgebungen, in denen alle Systeme jederzeit verfügbar sind, ist lediglich eine initiale Konfiguration der Systeme erforderlich. Eine eventuell vorhandene Dynamik in der Verfügbarkeit oder in der Adressierung der Webservices kann durch Netzwerkdienste wie UDDI oder DNS bewältigt werden. Besteht die Umgebung aber aus mobilen Systemen, müssen sich diese zum großen Teil selbst konfigurieren, da sich die URLs, unter denen die Webservices angesprochen werden können, je nach Verfügbarkeit der Systeme, ändern. Um diese Dynamik in der IT-Umgebung zu unterstützen, müssen die

beteiligten Systeme also stets aktuelle Informationen über die im Netzwerk verfügbaren Teilnehmer haben. Jedes mobile System muss wissen, welche anderen Teilnehmer verfügbar sind und welche Webservices diese bereitstellen. Eine Lösung für diese Anforderung ist im Kapitel 3: „WLAN Ad-Hoc Kommunikation“ beschrieben.

Die zweite Anforderung, mit der sich dieser Report befasst, ist die Umsetzung eines Store-Carry-Forward Mechanismus zur Übermittlung von Daten, falls keine direkte Kommunikationsmöglichkeit zwischen Sender und Empfänger zur Verfügung steht. Die Idee hierbei ist, dass die Daten von Maschine zu Maschine (hier Landmaschinen) übertragen werden, bis sie letztendlich das gewünschte Ziel erreichen. Hat eine Maschine keine Verbindung zu einer weiteren, so werden die Daten gespeichert und erst beim Zusammentreffen mit einer anderen Maschine abgeglichen.

Die Funktionsweise dieses Mechanismus wird an dem Beispielszenario „Hinderniswarnung“ gezeigt, welches ebenfalls von CLAAS eingebracht wurde. Bei der Hinderniswarnung geht es darum, dass der Fahrer einer Landmaschine gewarnt werden soll, wenn er auf ein Hindernis zufährt. Hierfür müssen möglichst alle erfassten Hindernisse, welche sich auf dem aktuell befahrenen Feld / Schlag befinden, bekannt sein. Der Store-Carry-Forward Mechanismus dient hierbei zum Abgleich der Hindernisinformationen unter den Maschinen auf einem Feld. Eine Lösung für diese Anforderung wird im Kapitel 4: „Store-Carry-Forward und Hinderniswarnung“ vorgestellt.

2 Verwendete Technologien

2.1 OSGi

OSGi [OSGI] ist eine Spezifikation einer java-basierten dynamischen Softwareplattform, welche eine Modularisierung und Verwaltung der Anwendungen und Dienste durch ein Komponentenmodell ermöglicht. Die Komponenten (auch „Bundles“ genannt) einer OSGi Umgebung können dynamisch gestoppt, gestartet, installiert und deinstalliert werden, ohne die OSGi Umgebung in Gänze herunterfahren oder neu starten zu müssen.

Die OSGi-Plattform setzt eine Java Virtual Machine (JVM) voraus und bietet darauf aufbauend das OSGi-Programmiergerüst.

2.1.1 Service Registry

Da die Service Registry für die in diesem Report vorgestellten Mechanismen von elementarer Bedeutung ist, wird diese im Folgenden etwas genauer beschrieben.

Die OSGi Service Registry verwaltet die durch die Bundles publizierten Services. Hier können Bundles zum einen ihre eigenen Services anmelden und bereitstellen, zum anderen können sie aber auch Services, welche von anderen Bundles bereitgestellt werden, ausfindig machen und verwenden. Die folgende Grafik zeigt die Interaktion zwischen den beteiligten Komponenten:

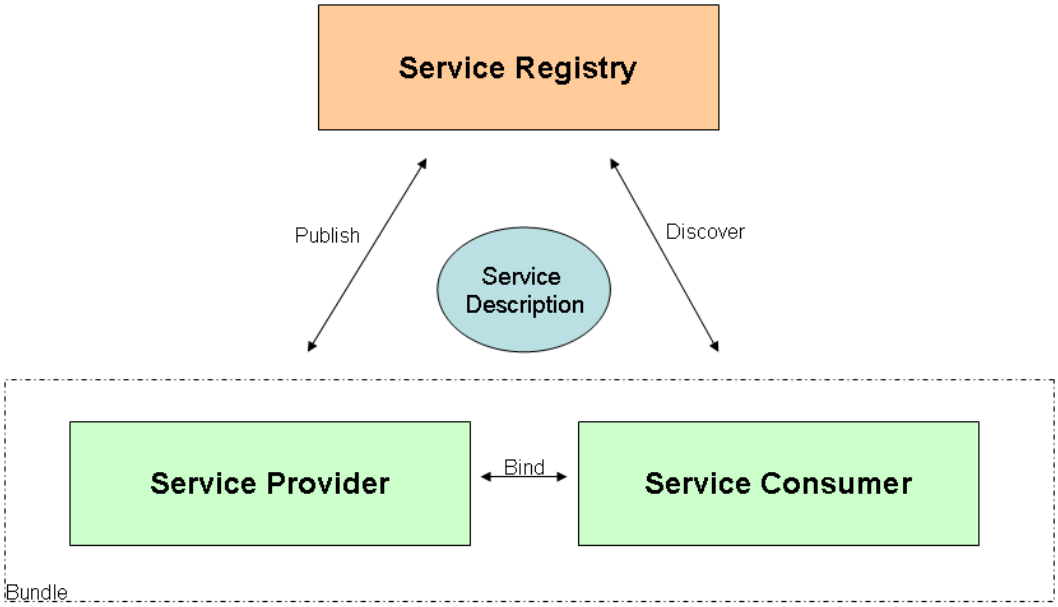

Abbildung 1: OSGi Service Registry

(Quelle: eigene Darstellung)

In diesem Fall befinden sich Service Provider und Service Requestor im gleichen Bundle. Der Service Provider publiziert seinen Service in der Service Registry. Der Service Requestor wartet auf die Verfügbarkeit eines bestimmten Services und benutzt diesen, sobald der Service verfügbar wird. Zur Ermittlung oder Benachrichtigung, wann ein Service verfügbar wird, stehen verschiedene Implementierungsmöglichkeiten zur Verfügung. Die in diesem Report vorgestellten Lösungen verwenden stets Declarative Services, um hochdynamisch auf Veränderungen in der Service Registry zu reagieren. Declarative Services erlauben eine sehr detaillierte Konfiguration der Bundles, bei der genau definiert werden kann, welche Services zwingend zum Start eines Bundles erforderlich sind und welche ggf. erst zu einem späteren Zeitpunkt verfügbar sein müssen. Sind alle notwendigen Services verfügbar, wird das Bundle gestartet. Kommen während der Laufzeit weitere (nicht zwingend erforderliche) Services hinzu, werden diese umgehend angebunden und können somit verwendet werden. Andersherum werden die Services auch umgehend wieder abgekoppelt, falls sie nicht mehr verfügbar sind. Somit eignet sich die Declarative Services Technik besonders für dynamische Umgebungen, in denen die Verfügbarkeit von Services schwer vorhersagbar und unzuverlässig ist.

2.1.2 Eclipse Equinox

Equinox [EQUINOX] ist ein von der Eclipse Foundation entwickelte Implementierung der OSGi-Spezifikation (R4, core framework specification), welche die Basis der integrierten Entwicklungsumgebung Eclipse bildet. Im Rahmen von R2B wird die Standalone Implementierung von Equinox als OSGi-Runtime Umgebung verwendet.

2.2 JGroups

JGroups [JGROUPS] ist ein Toolkit für zuverlässige Multicasting-Kommunikation. In erster Linie findet eine Kommunikation in Form von Nachrichten statt, die unter den Teilnehmern verschickt werden. Damit die Teilnehmer untereinander kommunizieren können, müssen diese zunächst einen virtuellen Kommunikationskanal erzeugen. An diesem Kanal können sich beliebig viele Teilnehmer anmelden. Wird eine Nachricht eines Teilnehmers in den Kanal geschickt, so wird diese automatisch allen angemeldeten Teilnehmern zugestellt (sofern die Nachricht nicht explizit als Unicast-Nachricht verschickt wird).

Sowohl ein explizites Abmelden von einem Kommunikationskanal, wie auch das Detektieren eines nicht mehr vorhandenen Kommunikationspartners (crashed partner) werden unterstützt.

Neben dieser Form der Gruppenkommunikation wird zur Umsetzung des Store-Carry-Forward Mechanismus noch ein weiteres Feature von JGroups in R2B verwendet. JGroups bietet verschiedene Möglichkeiten innerhalb der Kommunikationsgruppe eine Art virtuellen Speicher zu erzeugen, in den jeder Teilnehmer Informationen einstellen kann und deren Informationen auf allen oder mehreren Teilnehmern repliziert werden.

2.3 Apache Derby

Als Datenbank wird die unter der Apache License lizenzierte Datenbanksoftware Derby verwendet. Die bestehende Software wurde in ein OSGi Bundle eingebettet, so dass diese innerhalb der OSGi Umgebung lauffähig ist.

3 WLAN Ad-Hoc Kommunikation

3.1 Anwendungsszenario

Wenn sich Landmaschinen auf einem Feld begegnen bzw. nah genug beieinander fahren, um per WLAN kommunizieren zu können, sollen diese in der Lage sein, Webservices auf den jeweils anderen Maschinen aufzurufen. Ein Webservice Aufruf könnte zum Beispiel den Start eines bestimmten Prozesses auf der anderen Maschine bewirken oder eine Bereitstellung von beliebigen Informationen zu bestimmten Prozessschritten ermöglichen.

3.2 Ad-hoc Informationsaustausch

Um eine serviceorientierte Architektur, deren Kommunikation auf Webservices basiert, herzustellen, müssen die beteiligten mobilen Systeme zu jeder Zeit exakt wissen, welche anderen Systeme sich aktuell im Netzwerk befinden. Sie müssen neben der Adressierungsinformationen auch Kenntnisse über die auf den Systemen verfügbaren Webservices haben, um eine zielgerichtete Interaktion der Systeme untereinander zu ermöglichen.

Moderne WLAN-Karten können –sofern sie im ad-hoc Modus betrieben werden– selbstständig Verbindungen untereinander aufbauen, sobald sich Systeme mit gleich konfigurierten Netzwerkeigenschaften in erreichbarer Nähe befinden. Die Hardware deckt somit die Kommunikation bis einschließlich Netzwerkschicht 3 (hier: IP Adresse) des ISO/OSI Modells ab.

Die hier vorgestellte Softwarelösung setzt nun auf diese durch die Hardware hergestellte Basiskommunikation auf. Sie ist in der Lage, zeitnah eine hergestellte Basiskommunikation zu erkennen und erweitert diese um einen Multicasting Kommunikationskanal, über welchen die beteiligten Systeme beliebige Informationen austauschen können. Wird eine Information in den Kanal gesendet, so erreicht diese alle Teilnehmer, die sich aktuell an dem Kanal angemeldet haben. Die durch JGroups bereitgestellten Reliable Messaging Mechanismen stellen eine zuverlässige Nachrichtenübermittlung sicher.

3.2.1 Implementierung im Detail

Die folgende Grafik zeigt den Ablauf des Verbindungsaufbaus und den anschließenden Informationsaustausch:

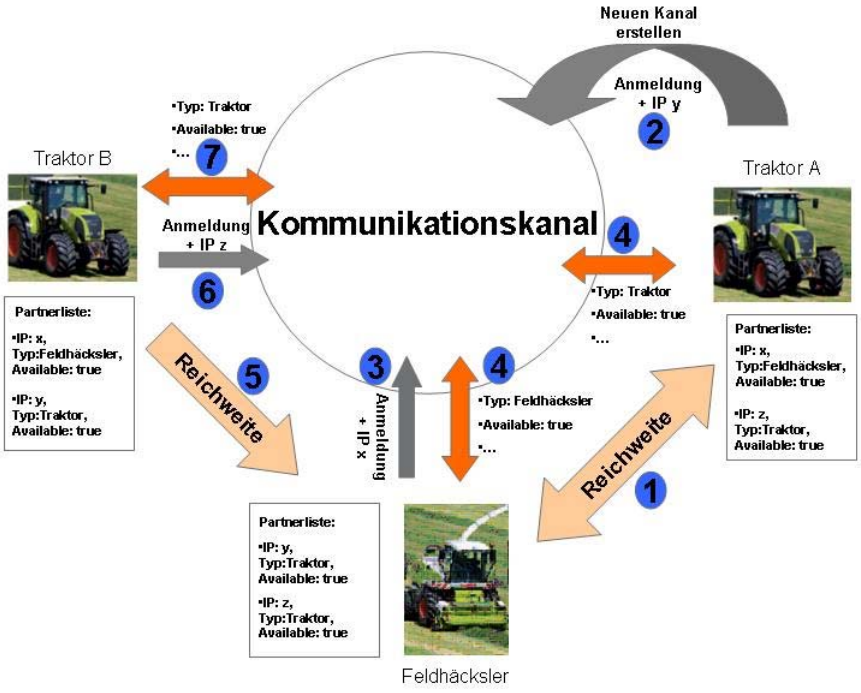

Abbildung 2: WLAN ad-hoc Kommunikation

(Quelle: eigene Darstellung)

Erläuterungen zur Grafik:

Zu 1: Der Feldhäcksler und Traktor A fahren nah genug beieinander, so dass sie sich in Funkreichweite der WLAN-Module befinden.

Zu 2: Einer der beiden Partner (der schnellere), hier Traktor A erzeugt einen neuen JGroups Kommunikationskanal (Multicasting).

Zu 3: Der Feldhäcksler meldet sich an dem Kommunikationskanal an.

Zu 4: Beide Systeme senden Informationen in den Kanal. In diesem Szenario wird der Typ der eigenen Maschine und ein Status verschickt. Die jeweils andere Maschine bekommt die Nachricht und kann neben den Nachrichteninhalt (Typ, Status) auch die IP Adresse des Absenders ermitteln. Beide Maschinen wissen nun, dass eine weitere Maschine in Reichweite ist, von welchem Typ diese ist und welche IP Adresse und welchen Status sie momentan hat.

Da im R2B Szenario alle Maschinen eines Typs die gleichen (Web-) Services zur Verfügung stellt, können die Maschinen anhand der Typdaten der Kommunikationspartner ermitteln, welche Services diese bereitstellen. Die beiden Maschinen (Feldhäcksler und Traktor A) haben nun alle benötigten Informationen, um über Webservices kommunizieren zu können.

Zu 5: Anschließend kommt ein weiterer Teilnehmer (Traktor B) in WLAN Funkreichweite der beiden Maschinen.

Zu 6: Die Software auf dem Traktor B erkennt, dass ein WLAN Netzwerk verfügbar ist und meldet sich am bereits bestehenden Kommunikationskanal an.

Zu 7: Anschließend verschickt auch Traktor B Informationen über das eigene System in den Kommunikationskanal. Gleichzeitig bekommt Traktor B auch von den bereits vorhandenen Maschinen (Feldhäcksler und Traktor) deren Informationen zugeschickt. Die weitere Verarbeitung verläuft analog zu der Beschreibung unter Punkt 4. Alle Teilnehmer wissen nun, welche weiteren Maschinen sich aktuell im Netzwerk befinden und haben alle notwendigen Informationen zur Verfügung, um über Webservices miteinander zu kommunizieren.

Innerhalb des Kommunikationskanals wird die Verfügbarkeit der Teilnehmer regelmäßig überprüft. Verlässt ein Teilnehmer die Reichweite der übrigen, so bekommen diese eine Meldung, dass die Maschine nicht mehr im Netzwerk vorhanden ist und können diese aus ihrer Kommunikationspartnerliste entfernen. Die Maschine, die die Reichweite verlassen hat, bemerkt selbstständig, dass kein Netzwerkpartner mehr verfügbar ist und stellt die Kommunikation ein.

Erkennung der Verfügbarkeit weiterer Maschinen

Die WLAN Netzwerkkarten stellen im ad-hoc Modus automatisch eine Verbindung her, sobald sich ein gleich konfiguriertes System in Reichweite befindet.

Zur regelmäßigen Überprüfung, ob eine Verbindung besteht, wird ein (ConnectionDetector-)Thread gestartet, der periodisch die Netzwerkeigenschaften des lokalen Systems überprüft. Erkennt dieser, dass eine Verbindung hergestellt wurde, wird ein OSGi Service in die Service Registry der OSGi Umgebung publiziert, wobei der Typ des Service in Form

einer Java Schnittstellenbeschreibung definiert wird (hier: ConnectionDetector).

Erzeugung / Anmeldung Kommunikationskanal

Eine weitere OSGi Komponente (WlanCommunicationService) horcht auf einen Service in der Service Registry, der die Schnittstellenbeschreibung ConnectionDetector implementiert. Das Vorhandensein des ConnectionDetectorServices ist zum Starten des WlanCommunicationService zwingend erforderlich. Sobald ein ConnectionDetectorService vorhanden ist, wird der WlanCommunicationService gestartet und eröffnet automatisch einen JGroups Kanal. Existiert dieser Kanal bereits, z.B. weil bereits mehrere Maschinen miteinander kommunizieren, so meldet sich das System an den bestehenden Kanal an. Eine Erzeugung eines neuen Kanals ist in diesem Fall nicht erforderlich.

Informationsaustausch

Nachdem ein neuer Kommunikationskanal eröffnet wurde oder sich ein weiterer Teilnehmer am Kanal angemeldet hat, tauschen alle Partner Informationen gegenseitig aus. Jeder Teilnehmer schickt eine Nachricht in den Kanal, welche aktuelle Informationen (hier: IP Adresse, Systemtyp, Status) über das eigene System enthält. Diese Nachrichten werden per Multicasting an alle anderen Teilnehmer versendet.
Des Weiteren führt jeder Teilnehmer eine lokale Partnerliste, in der die Informationen über alle verfügbaren Kommunikationspartner gehalten werden.

Kommunikation über Webservices

Die Partnerliste enthält Informationen über IP Adressen, Maschinentyp und Status der momentan erreichbaren Kommunikationspartner. Des Weiteren wird in der lokalen Datenbank geführt, welcher Maschinentyp welche Webservices bereitstellt. Anhand dieser Daten lässt sich daher ermitteln, welcher momentan vorhandene Kommunikationspartner über welche Webservices angesprochen werden kann. Aus den IP-Adressen und den Serviceinformationen können die Maschinen nun die kompletten URL Zeichenketten ermitteln, unter denen sie per Webservice Aufruf kommunizieren können.

Abmeldung / Abbau Kommunikationskanal

Verlässt ein Kommunikationspartner die Reichweite der übrigen Teilnehmer, so wird die Verbindung der WLAN Netzwerkkarte getrennt. Der ConnectionDetector-Thread erkennt, dass keine WLAN Verbindung mehr zur Verfügung steht und zieht die Publikation des Services aus der OSGi Registry zurück. Da das Vorhandensein dieses Services für den WlanCommunicationService zwingend erforderlich ist, wird auch dieser Service gestoppt, wodurch der JGroups Kommunikationskanal geschlossen wird.

Die übrigen Teilnehmer des Kanals können nach wie vor untereinander kommunizieren. Da der Zustand aller Teilnehmer in regelmäßigen Abständen überprüft wird, erkennen diese, dass ein Teilnehmer den Kanal verlassen hat und entfernen den entsprechenden Eintrag aus ihrer Partnerliste.

4 Store-Carry-Forward und Hinderniswarnung

4.1 Anwendungsszenario Hinderniswarnung

Ein weiteres Szenario im R2B Projekt ist die Hinderniswarnung. Hierbei soll der Fahrer einer Maschine gewarnt werden, wenn sich ein Hindernis in seiner näheren Umgebung (Bspl: Umkreis 20m) befindet. Damit der Fahrer von seiner Maschine gewarnt werden kann, muss diese stets aktuelle Hindernisinformationen zur Verfügung haben. Die Hindernisinformationen werden in einem Farmmanagementsystem (nachfolgend auch Backend bezeichnet) des Farmers gespeichert und bei vorhandener Konnektivität auf die Maschinen überspielt. Da die Informationen manuell, nach einer Überprüfung und Zustimmung eines Administrators in die Datenbank im Farmmanagementsystem aufgenommen wurden, werden diese Hindernisse als „validiert“ bezeichnet. Neben diesen validierten Hindernissen, ist es auch erforderlich, weitere (neue) Hindernisdaten auf dem Feld durch den Fahrer aufzunehmen und diese Information anderen Maschinen zeitnah zur Verfügung zu stellen. Da diese Informationen noch nicht überprüft wurden, werden diese Hindernisse als „nicht validiert“ bezeichnet. Sofern keine Satelliten- oder GPRS Kommunikation vorhanden ist, müssen diese Informationen per WLAN von Maschine zu Maschine weitergegeben werden, welches als Store-Carry-Forward (SCF) Prinzip bezeichnet wird. Hierbei werden die Informationen lokal gespeichert (store), mit der Maschine mitgenommen (carry) und an weitere Maschinen versendet, sobald diese in Kommunikationsreichweite sind (forward). Das Ziel ist zum einen, dass diese Informationen am Backend ankommen und dort, nach einer Überprüfung, entweder in die validierte Datenbank aufgenommen werden, oder eine zeitnahe Räumung des Hindernisses geplant werden kann. Zum anderen soll die Hindernisinformation an möglichst viele Maschinen in der näheren Umgebung verteilt werden, so dass deren Fahrer ebenfalls gewarnt werden, wenn sie auf dieses Hindernis treffen.

Um eine Überlastung des Netzwerks zu verhindern, müssen die per SCF übertragenen Informationen regelmäßig aus dem Netzwerk entfernt werden.

4.2 Store-Carry-Forward Mechanismus

Grundlage des Store-Carry-Forward Mechanismus ist ein mobiles ad-hoc Netzwerk (MANET), also ein spontaner Aufbau eines Netzwerks zwischen mobilen Maschinen, abhängig von der Kommunikationsreichweite. Aufbauend

auf dieser Kommunikation wird ein virtueller Speicher angelegt, in den alle verbundenen Maschinen ihre Daten einspeisen können und deren Inhalte auf alle Maschinen repliziert werden. Wird eine Information in den Speicher eingestellt, ist diese somit auf allen verbundenen Teilnehmern verfügbar. Jede Information wird zudem mit einer Lebenszeit (Time-To-Live (TTL)) versehen, nach deren Ablauf die Information wieder aus dem Speicher entfernt wird.

4.2.1 Implementierung im Detail

Hindernisinformationen

Zur Speicherung von Hindernisinformationen werden auf allen Systemen (Maschinen und Farmmanagement) zwei Datenbanktabellen geführt. Eine Tabelle enthält die bereits überprüften Hindernisinformationen („validiert“) und die andere die neu eingegebenen und noch nicht überprüften („nicht validiert“).

In beiden Tabellen werden die Hindernisse in Form von Polygonen gespeichert, wodurch eine Abbildung von beliebig komplexen Hindernissen ermöglicht wird. Gespeichert werden jeweils die GPS Koordinaten der Eckpunkte des Polygons. Für jedes Hindernis wird außerdem die ID des Schlages geführt, auf dem sich dieses Hindernis befindet. So können die Maschinen vor und während des Einsatzes gezielt die Hindernisinformationen bekommen, die sie zur Bearbeitung ihres Schlages benötigen.

Eingabe neuer Hindernisse durch den Fahrer

Erkennt ein Fahrer einer Maschine ein noch nicht erfasstes Hindernis, so kann dieser das Hindernis über eine grafische Benutzeroberfläche (GUI) eingeben. Um eine möglichst schnelle und einfache Eingabe eines Hindernisses zu ermöglichen, ist die Angabe von Richtung, Entfernung und Radius (jeweils Schätzwerte) erforderlich. Richtung und Entfernung beziehen sich hierbei jeweils auf den Mittelpunkt des Hindernisses. Ein Algorithmus berechnet dann aus Richtung, Entfernung und den aktuellen GPS Koordinaten (inkl. Fahrtrichtung der Maschine) die (geschätzte) Position des Hindernismittelpunkts. Da die Hindernisinformationen in der Datenbank als Polygone gespeichert werden, wird anhand des ermittelten Hindernismittelpunkts und dem eingegebenen Radius ein Achteck berechnet, deren acht Eckpunkte sich im eingegebenen Radius um den Mittelpunkt

befinden. Diese Punkte werden anschließend in der lokalen Datenbank gespeichert.

Die folgende Grafik zeigt ein Beispiel eines aus Mittelpunkt und Radius berechneten Polygons:

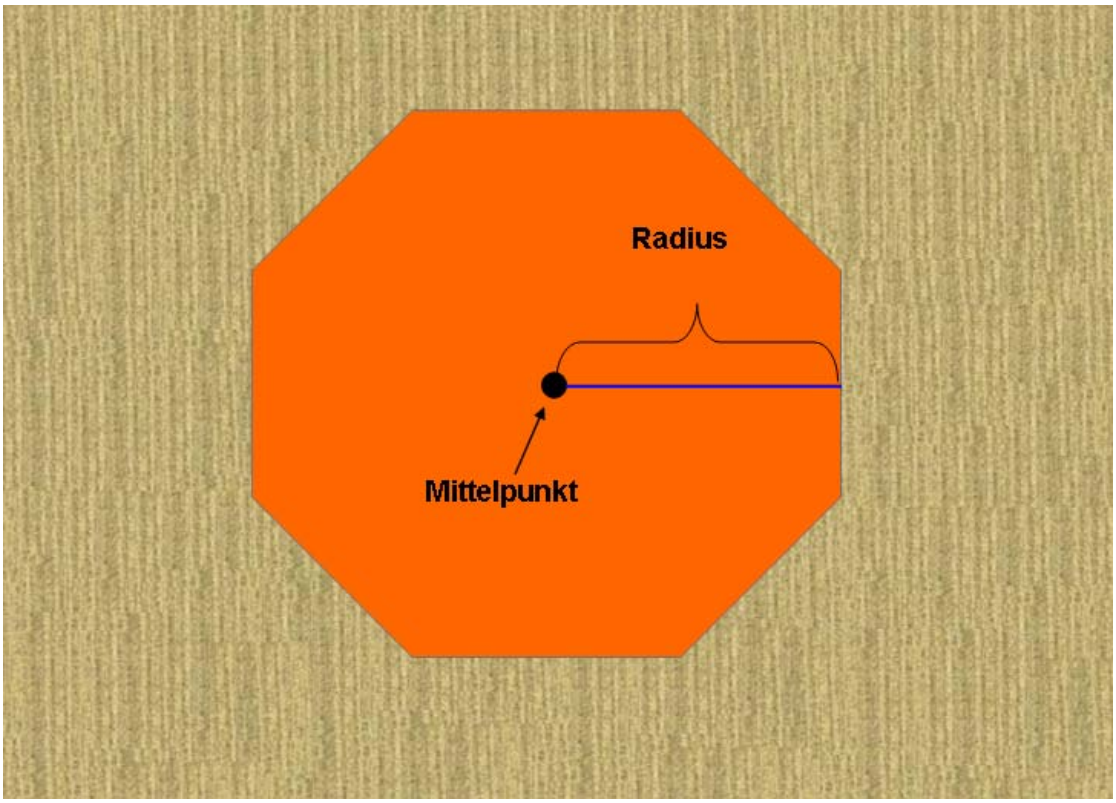

Abbildung 3: Berechnetes Hindernis (Achteck)

(Quelle: eigene Darstellung)

Publikation der Hindernisinformation

Wurde ein neues Hindernis durch einen Fahrer erkannt und eingegeben, so muss diese Information anschließend weiteren Maschinen weitergegeben werden, die ggf. auf dieses Hindernis treffen könnten. Zum anderen muss die Information in die Datenbank des Farmmanagement-Systems eingetragen werden, so dass das Hindernis überprüft und entweder geräumt oder als dauerhaftes Hindernis aufgenommen werden kann.

Diese Weitergabe der neu eingegebenen Hindernisinformationen wird mittels des Store-Carry-Forward Mechanismus durchgeführt. Begegnen sich zwei oder mehrere Maschinen, so beginnt zunächst die in Kapitel 3 beschriebene ad-hoc Kommunikation. Der Kommunikationskanal wird anschließend durch einen virtuellen Speicher erweitert (in Form einer JGroups ReplicatedHashMap). Fügt eine Maschine eine Information in diesen Speicher

ein, so wird diese Information auf alle angemeldeten Teilnehmer repliziert. Angemeldet sind automatisch alle Teilnehmer, die sich auch an dem Kommunikationskanal angemeldet haben. Die Informationen in dem Speicher sind somit stets auf allen angemeldeten Teilnehmern verfügbar. Um das zu übertragene Datenvolumen in der Kommunikation zwischen den Maschinen möglichst gering zu halten, werden nicht die vollständigen Polygondaten in den Speicher gestellt (GPS-Koordinaten der Eckpunkte), sondern lediglich der Mittelpunkt und der Radius des Polygons. Mittels dieser Informationen können die Maschinen selbständig unter Verwendung des oben beschriebenen Algorithmus die acht Eckpunkte des Polygons berechnen.

Die folgende Grafik zeigt beispielhaft, wie eine Information von Maschine zu Maschine weitergetragen wird:

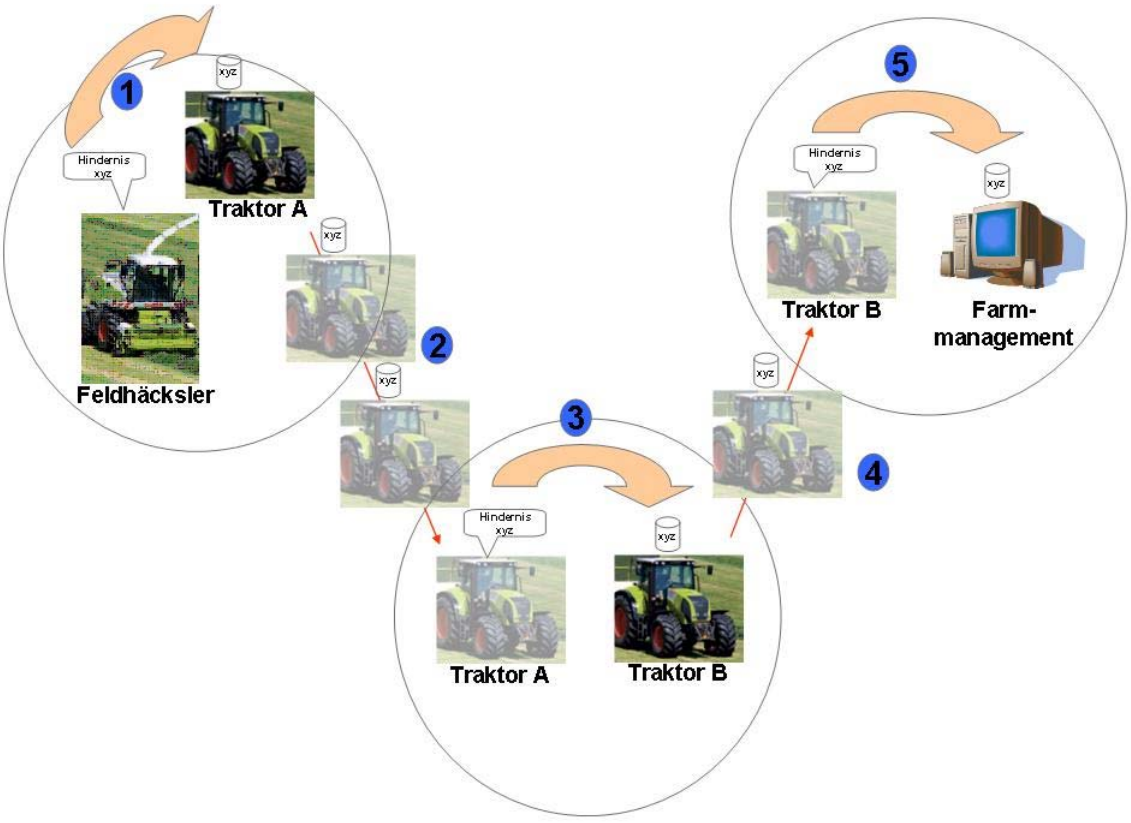

Abbildung 4: Store-Carry-Forward

(Quelle: eigene Darstellung)

Erläuterungen zur Grafik:

Zu 1: Zunächst begegnen sich der Feldhäcksler und Traktor A, wobei begegnen hier bedeutet, dass sie nah genug beieinander sind, um per WLAN kommunizieren zu können. Der Feldhäcksler hat hier Informationen aufgenommen (z.B. über ein neues

Hindernis) und gibt diese an den Traktor A weiter. Zudem werden die Informationen lokal auf dem Feldhäcksler sowie auf dem Traktor A gespeichert und mit einer TTL versehen.

Zu 2: Anschließend verlässt Traktor A die (WLAN-)Reichweite des Feldhäckslers, trägt aber die gespeicherte Information mit sich.

Zu 3: Beim Aufeinandertreffen von Traktor A und Traktor B wird die Information umgehend an Traktor B übermittelt, so dass auch dieser die Daten zur Verfügung hat. Dieser speichert ebenfalls die Information lokal ab und trägt sie fortan weiter.

Zu 4: Traktor B verlässt die Reichweite von Traktor A, behält aber die Information gespeichert.

Zu 5: Kommt Traktor B am Heimat-Hof an, wird über identische Mechanismen ein Netzwerk zum Farmmanagementsystem hergestellt und die Information auch dorthin übertragen.

Auf den hier beschriebenen Weg gelangt die Information zu allen Maschinen, die mit einer Maschine in Kontakt kommen, welche die Information bereits gespeichert hat. Nach einiger Zeit hat die Information mit hoher Wahrscheinlichkeit alle Maschinen auf dem Feld erreicht.

Das zweite Ziel, die Übertragung der Information an das zentrale Farmmanagementsystem, kann auf zweierlei Weise erreicht werden. Besteht eine direkte Verbindung zwischen Maschine und Farmmanagementsystem (z.B. über GPRS, UMTS), können die Daten direkt übertragen werden. Ist das nicht der Fall gelangt die Information am Farmmanagementsystem an, sobald eine Maschine diese Information dorthin „trägt“, welches z. B. bei Abtankfahrzeugen sehr wahrscheinlich ist.

Ist der Zeitstempel, der durch die TTL beim erstmaligen Einstellen der Information in den Speicher vorgegeben wird, erreicht, so wird die Information aus dem Speicher entfernt. Auf allen Maschinen läuft ein Thread, der in regelmäßigen Abständen überprüft, ob die Informationen im lokalen Speicher beibehalten oder entfernt werden müssen und dann entsprechend handelt. Auf diese Weise wird die Information relativ zeitgleich auf allen Maschinen aus dem lokalen Speicher entfernt, unabhängig davon, ob sie momentan miteinander kommunizieren oder nicht.

Warnung des Fahrers

Zur Warnung des Fahrers, ob sich ein Hindernis in der Nähe befindet, werden zunächst alle Hindernisinformationen, welche zum aktuellen Schlag gehören, aus der validierten und der nicht validierten Hindernistabelle in einen lokalen Cache gespeichert. Beim Speichern in den Cache werden die Hindernisse um einen bestimmten Faktor (>1) gezoomt. Dieser Zoomfaktor wird berechnet aus dem Radius des Hindernisses (Abstand vom entferntesten Punkt zum Mittelpunkt) und dem eingestellten Warn-Schwellwert (Abstand zum Hindernis, ab dem gewarnt werden soll).

Während der Fahrt der Maschine wird nun permanent geprüft, ob sich die Maschine innerhalb eines dieser vergrößerten Polygone befindet. Ist das der Fall, wird der Fahrer über die Anzeige eines entsprechenden Warnhinweises auf einem Monitor gewarnt.

Das folgende Bild zeigt die Polygone (original und skaliert) und Beispielpositionen der Maschine:

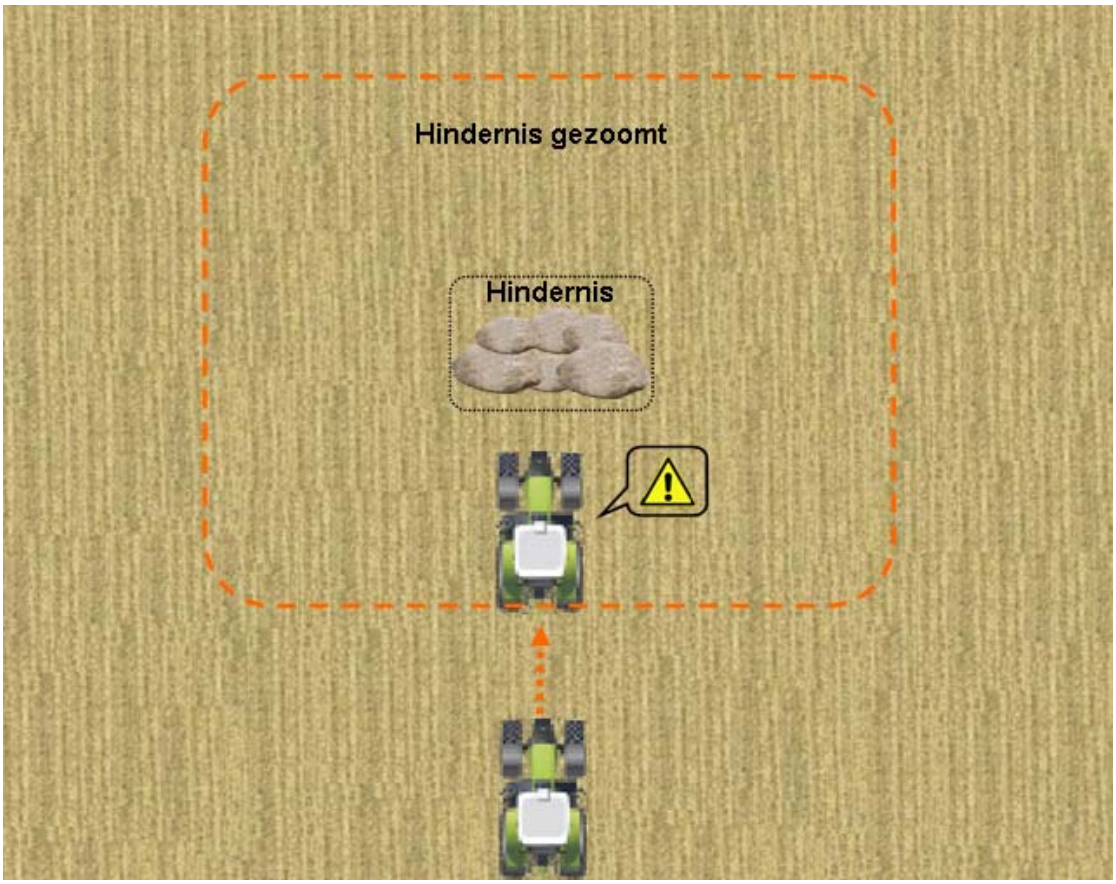

Abbildung 5: Hinderniswarnung

(Quelle: eigene Darstellung)

Das Polygon (schwarz umrandet) in der Mitte des Bildes bildet das Hindernis in Originalgröße ab. Das äußere (orange gestrichelt) ist das vergrößerte Polygon, welches in einem Cache auf der Maschine gespeichert wird. Ein Punkt-in-Polygon Algorithmus prüft nun permanent ab, ob sich die aktuelle Position der Maschine innerhalb des skalierten Hindernispolygons befindet. Ist das der Fall, bekommt der Fahrer eine Warnung auf dem Monitor angezeigt. Durch den hier gezeigten Mechanismus kann auf einfache Weise sichergestellt werden, dass der Fahrer in jedem Fall vor Erreichen des Hindernisses (Kollision) gewarnt wird. Der Schwellwert (in Metern), ab dem gewarnt werden soll, kann vom Fahrer festgelegt werden.

5 Fazit / Ausblick

In diesem Report wurden Lösungen im Bereich der Kommunikation von mobilen Teilnehmern vorgestellt, welche aus den Anforderungen im Projekt R2B hervorgegangen sind. Die Herausforderungen waren:

(1) Aufbau einer ad-hoc Kommunikation zwischen mobilen Teilnehmern / Maschinen

(2) Implementierung eines Store-Carry-Forward Mechanismus

Die hier vorgestellten Lösungen verwenden JGroups zur Multicasting-Kommunikation, welches in Form eines Bundles in die OSGi Umgebung Equinox integriert wurde. Um softwareseitig auf die Dynamik eines mobilen ad-hoc Netzwerkes (MANET) reagieren zu können, wurden Mechanismen der OSGi Umgebung, insbesondere Declarative Services, verwendet.

Die Lösungen sind so generisch gehalten, dass eine Übertragbarkeit auf verschiedene Anwendungsgebiete problemlos möglich ist.

So kann die Lösung für die ad-hoc Kommunikation grundsätzlich überall dort verwendet werden, wo mobile Systeme im Falle eines Zusammentreffens ad-hoc über Netzwerk kommunizieren und Daten austauschen müssen. Mobile Systeme können hierbei beliebige Fahrzeuge, aber auch Handys, PDAs oder Notebooks sein.

Um einen Einsatz in verschiedene Domänen zu ermöglichen ist der Netzwerktyp (hier: WLAN) durch ein beliebiges anderes Funknetzwerk ersetzbar. Des Weiteren können beliebige Datenformate und damit auch Inhalte ausgetauscht werden.

Zur Integration der mobilen Teilnehmer in einen Geschäftsprozess kann die Lösung noch um eine Schnittstellenschicht erweitert werden, welche die Daten, die über die ad-hoc Kommunikation ausgetauscht werden, einer Workflow-Engine (wie z.B. BPEL Engine) zur Verfügung stellt. Des Weiteren kann die Lösung in sofern erweitert werden, dass sie vollkommen unabhängig vom Kommunikationsmedium agiert. So wäre neben der WLAN Kommunikation auch eine Datenübertragung z.B. via Bluetooth, ZigBee, UWB oder andere Funknetze denkbar. Insbesondere Funknetze, welche sehr schnell ein ad-hoc Netzwerk aufspannen können, wie z.B. Car-WLAN könnten die hier verwendete klassische WLAN Technologie ersetzen oder ergänzen.

Der hier vorgestellte Store-Carry-Forward Mechanismus ermöglicht eine Datenübertragung, bei dem keine direkte Verbindung zwischen Sender und Empfänger besteht. Hierbei werden die Daten von den mobilen Teilnehmer über beliebig viele Hops vom Sender zum Empfänger „getragen“. Auch hier ist die Lösung so generisch gehalten, dass eine Anwendung in beliebigen Domänen möglich ist. Ein Zeitmechanismus regelt die Gültigkeitsdauer der Daten und entfernt Informationen aus dem Speicher, wenn der entsprechende Zeitstempel überschritten ist. Somit wird ein Überlaufen des Speichers auf den mobilen Teilnehmern verhindert, sowie ein Ping-Pong Effekt, bei dem ein Datensatz dauerhaft im Netzwerk verbleibt.

Das Problem, welches ein Store-Carry-Forward Mechanismus mit sich bringt, ist, dass weder die Zeit, wann die Information den Empfänger erreicht, exakt bestimmt werden kann, noch sichergestellt werden kann, dass die Information den Empfänger überhaupt erreicht. Um aber die Wahrscheinlichkeit einer zeitnahen Datenübertragung zum Empfänger zu erhöhen, könnte die Lösung um weitere intelligente Komponenten ergänzt werden. So ist zum Beispiel ein Ansatz die Systeme mit einem Gütefaktor zu versehen, welcher eine Bewertung über die Kommunikationsmöglichkeit eines Geräts darstellt. Dieser Gütefaktor müsste ständig neu berechnet werden, so dass stets die aktuelle Situation bestmöglich widergespiegelt wird. Bei den Landmaschinen in dem hier beschriebenen Szenario beispielsweise könnte dieser Gütefaktor Auskunft darüber geben, wie häufig eine Maschine mit dem Farmmanagementsystem kommuniziert. Die Informationen, welche das Farmmanagement erreichen sollen, könnten dann insbesondere denjenigen Maschinen weitergegeben werden, welche einen hohen Gütefaktor haben. Hierdurch würde zum einen die Wahrscheinlichkeit erhöht, dass die Daten zeitnah am Ziel ankommen, zum anderen würden die Speicher auf den übrigen Maschinen geschont.

Eine weitere Erweiterung zu der hier vorgestellten Lösung könnte der Aufbau einer Verbindung zwischen Sender und Empfänger sein, auch wenn sich Sender und Empfänger nicht direkt erreichen können. Hier könnten andere mobile Teilnehmer als Router agieren, um eine Netzwerkverbindung zwischen Sender und Empfänger zu ermöglichen. Ein Lösungsansatz hierfür könnte das M-Channel [M-Channel] Toolkit bereitstellen.

6 Referenzen / Glossar

R2B:	**Homepage, http://www.r2b-online.de**
BMWI :	**Homepage, http://www.bmwi.de**
SIMOBIT:	**Homepage, http://www.simobit.de/**
CLAAS:	**Homepage, http://www.claas.de**
CADsys:	**Homepage, http://www.cadsys.de**
OSGI:	**Homepage, http://www.osgi.org/**
EQUINOX:	**Homepage, http://www.eclipse.org/equinox**
JGroups:	**Homepage, http://www.jgroups.org/**
Apache Derby:	**Homepage, http://db.apache.org/derby/**
M-Channel:	**Homepage, http://ast-deim.urv.cat/mchannel/**

SOA:	**Serviceorientierte Architektur**
URL:	**Uniform Resource Locator**
UDDI:	**Universal Description, Discovery and Integration**
DNS:	**Domain Name System**
Schlag:	**Feldstück**
MANET:	**Mobile ad-hoc Network**
UWB:	**Ultra Wide band**
WLAN:	**Wireless Local Area Network**
GUI:	**Graphical User Interface**