Publish/Subscribe Middleware für Selbstorganisierende Drahtlose Multi-Hop-Netzwerke

André Herms[1] and Michael Schulze[2]

[1] Institut für Verteilte Systeme
Arbeitsgruppe Echtzeitsysteme und Kommunikation
Universität Magdeburg
aherms@ivs.cs.uni-magdeburg.de
[2] Institut für Verteilte Systeme
Arbeitsgruppe Eingebettete Systeme und Betriebssysteme
Universität Magdeburg
mschulze@ivs.cs.uni-magdeburg.de

Zusammenfassung Das Routing in Wireless Mesh Networks hat einen dynamischen und selbstorganisierenden Charakter. Allerdings erfolgt die Kommunikation auf Anwendungsebene adressbasiert, was eine statische Zuordnung von Kommunikationspartnern erfordert. Dies wirkt der dynamischen Struktur leider entgegen. In dieser Arbeit beschreiben wir die Verbindung der Publish/Subscribe-Middleware COSMIC mit dem Routing-Protokoll AWDS. Durch Verwendung von COSMIC können Anwendungen inhaltsbasiert kommunizieren und sind nicht mehr auf die Kenntnis von Adressen angewiesen. Es wird das Konzept beschrieben und mittels Simulation und realen Messungen evaluiert. Hierbei zeigen wir, dass Wireless Mesh Networks geeignet sind, um in einer P/S-Infrastruktur eingesetzt zu werden.

1 Einleitung

Wireless Mesh Networks (WMNs) stellen eine flexible, effektive und kostengünstige Möglichkeit zur Vernetzung größerer Areale dar, da auf die Installation einer drahtgebundenen Infrastruktur verzichtet werden kann. Durch die Selbstorganisation der Netzwerkinfrastruktur ist nur ein minimaler administrativer Aufwand nötig. Ein mögliches Anwendungsgebiet ist beispielsweise die Ferndiagnose und Überwachung großer industrieller Anlagen. Diese verfügen über diverse Sensoren/Aktoren, welche zur Regelung dienen und meistens über Feldbusse angebunden sind. Eine direkte Kommunikation aus einem WMN ist jedoch schwierig, da bereits eine einheitliche Adressierung in einer solchen stark heterogenen Infrastruktur kaum zu erreichen ist.

Eine Lösung ist die Verwendung von Publish/Subscribe-Kommunikation die auf inhaltsbasierte Adressierung baut. Zur Verbindung der stark unterschiedlichen Netzwerke kann eine Middleware dienen, wie das hier genutzte COSMIC

(**CO**operating **SM**art dev**IC**es). Die folgende Arbeit beschreibt einen aktuellen Forschungsansatz zur Verbindung von Wireless Mesh Networks mit Publish/Subscribe-Kommunikation.

Die Arbeit ist wie folgt gegliedert. In Abschnitt 2 wird die COSMIC Middleware vorgestellt und in Abschnitt 3 das genutzte AWDS-Routing (**A**d-Hoc **W**ireless **D**istribution **S**ervice). Der Abschnitt 4 beschreibt die Integration der beiden Systeme. Eine Evaluierung durch Messungen und Simulation ist in Abschnitt 5 zu finden. Wir enden mit den verwandten Arbeiten und einem Fazit mit Ausblick.

2 COSMIC Middleware

Die Middleware COSMIC, beschrieben in [KB02, KBMP03, SZ07], bietet ein ereignisbasiertes Kommunikationsmodell entsprechend dem Publish/Subscribe-Konzept an. Im Gegensatz zur adressbasierten Kommunikation wird ein inhaltsbasiertes Schema verwendet und Ereignisse stellen die Hauptabstraktion des Interaktionsmodells dar. Publisher und Subscriber spezifizieren die Art des Ereignisses, welches sie produzieren beziehungsweise konsumieren. Folglich offeriert COSMIC eine spontane und dynamische n-zu-m Kommunikation, die keine impliziten Annahmen bezüglich Synchronität der Ereignisse macht. Eine asynchrone Kommunikationform verhindert überdies Kontrollflussabhängigkeiten und sie führt infolgedessen zur Autonomie der Teilnehmer.

Ereignisse als Träger der getypten Information können im Allgemeinen auf zwei verschiedene Weisen entstehen. Auf der einen Seite kann ein Ereignis spontan durch Hardware erzeugt werden, z.B. ausgelöst durch Detektion an einem Sensor. Dies bedeutet, die Umwelt ist der Stimulus für das Ereignis. Ein Ereignis kann auf der anderen Seite aber auch periodisch durch eine Uhr initiiert werden, um beispielsweise den Status einer Variablen innerhalb eines Knotens kontinuierlich im System zu verbreiten. Ein COSMIC-Ereignis besteht dabei aus drei Teilen:

- einem *Subjekt*, repräsentiert durch einen eindeutigen 64-Bit-Bezeichner (Unique Identifier), welcher den Inhalt charakterisiert,
- den *Inhalt* beziehungsweise die Daten selbst, z.B. den Wert einer Distanzmessung oder eine Temperatur und
- zusätzliche *Attribute* (z.B. Sensorposition, Kontext, Qualität), welche optional sind

Das Subjekt eines Ereignisses hat nicht nur eine Bedeutung innerhalb eines Teilnetzes, sondern es ist eindeutig über verschiedene Netzwerke. Die Subjekte bilden somit einen global eindeutigen Adressraum. Zwischen verschiedenen Teilnetzen vermitteln Gateways, welche die Eindeutigkeit eines Subjekts nutzen, um den Informationsfluss zu filtern. Wenn ein Subjekt beispielsweise nicht außerhalb des spezifischen Teilnetzes angefordert wurde, wird das entsprechende Ereignis auch nicht durch ein Gateway in das andere Netz propagiert. Der prinzipielle Aufbau

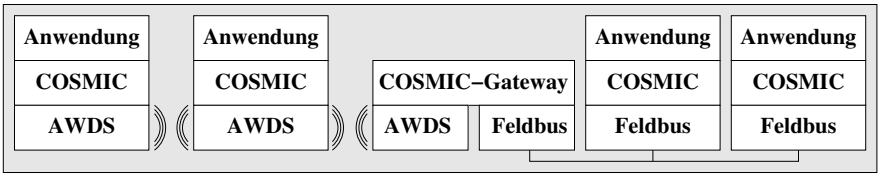


Abbildung 1. Integration von AWDS-Routing und COSMIC Middleware

eines verteilten Systems, welches COSMIC verwendet und über verschiedene Netzwerke kommuniziert, ist in Abbildung 1 zu sehen.

Neben den beschriebenen Ereignissen nutzt COSMIC Ereigniskanäle als Abstraktion für die Ereignisübertragung. Ein Ereigniskanal ist unidirektional und die Definition eines Ereigniskanals ist dem eines Ereignisses sehr ähnlich und gliedert sich ebenfalls in drei Teile:

- das *Subjekt*, ist das selbe wie das des korrespondierenden Ereignisses,
- die *Attribute*, Disseminationseigenschaften wie Periode, Frist, maximale Latenz, Zuverlässigkeit
- und *Handler*, für den Subscriber den Ausnahmehandler, falls ein Ereignis nicht rechtzeitig entsprechend der spezifizierten Attribute ankommt bzw. den Benachrichtigungshandler, wenn ein Ereignis eingetroffen ist. Für einen Publisher existiert nur der Ausnahmehandler, welcher aktiviert wird, wenn das Ereignis nicht entsprechend der Attribute veröffentlicht wurde.

Ereigniskanäle wurden nicht nur eingeführt, um der Anwendung die Spezifikation von Übertragungseigenschaften zu ermöglichen, sondern sie bieten der Middleware überdies die Möglichkeit der Reservierung der notwendigen lokalen sowie netzwerkseitigen Ressourcen, sobald ein Ereigniskanal erzeugt wird. Abhängig von den zeitlichen Bedingung oder der Wichtigkeit werden Ereigniskanäle in drei Qualitätsstufen als *harte*, *weiche* und *keine Echtzeit* eingeteilt. Außer den Ressourcen wird bei der Erzeugung eines Ereigniskanals auch das Binden der Subjekts an eine spezifische Netzwerkadresse sowie im Falle von harter Echtzeit Bandbreitenanforderungen durchgeführt. Dies geschieht jedoch vollständig transparent für die Anwendung und COSMIC verbirgt somit die Heterogenität des unterliegenden Netzwerkes vollständig.

Beim Einsatz der Middleware läuft typischerweise auf jedem Knoten eine Instanz von COSMIC, die sich beim Start automatisch in das bestehende Netzwerk integriert und die von der Anwendung benutzten Ereigniskanäle einrichtet. COSMIC wurde speziell, jedoch nicht ausschließlich, für den Einsatz und die Kooperation zwischen intelligenten Sensoren und Aktoren entwickelt. Die Middleware ist auf einer breiten Hardwarebasis angefangen von 8-Bit Mikrocontrollern (Atmel AVR, HC08), über 16-Bit Mikrocontroller (Siemens C167) jeweils nativ bis hin zu 32-Bit Systemen unter Linux und Windows einsetzbar und kann über verschiedene Kommunikationsnetze z.B. CAN (Controller Area Network [Rob91], TCP/IP, ZigBee [Zig03] interagieren.

Ein weiterer Schritt zur Verbreiterung der Anwendbarkeit stellt die hier vorgestellte Verbindung mit dem AWDS-Routing als konkrete Netzwerktechnologie dar, welche im Folgenden näher beschrieben wird.

3 AWDS Routing

AWDS (siehe auch [AWD08]) ist eine Open-Source-Implementierung eines Wireless Mesh Routing. Das Routing-Protokoll ist ein proaktives Link-State-Routing auf Schicht 2.

Die Software bietet zwei grundlegende Kommunikationsmodi – Versenden von Unicast-Nachrichten und Fluten. Unicast-Nachrichten werden anhand der Routing-Tabellen jeder Station bis zum Ziel weitergeleitet. Beim Fluten versendet eine Station die Nachricht als Broadcast-Paket. Alle empfangenden Stationen wiederholen das Paket genau einmal. Auf diese Weise erreicht das Paket jede Station. Jedes Flut-Paket wird genau n-mal wiederholt, wenn n die Anzahl der Stationen im Netz ist. Die Reichweite kann aber durch setzen eines TTL-Feldes (Time to Live) beschränkt werden. Jede Station decrementiert den entsprechenden Wert beim Weiterleiten und verwirft das Paket bei TTL=0.

Die Implementierung lässt sich auf unterschiedliche Art und Weise mit einer Anwendung kombinieren. Der übliche Weg ist der Einsatz von AWDS als Routing-Dienst. Dabei stellt jede Instanz auf den einzelnen Rechnern eine virtuelle Netzwerkschnittstelle bereit. Pakete, die vom Betriebsystem über diese Schnittstellen versandt werden, verpackt der Routing-Dienst in AWDS-Nutzdaten und leitet sie an das entsprechende Ziel weiter. Auf diese Weise verhalten sich alle virtuellen Netzwerkschnittstellen wie in einem gemeinsamen Schicht-2-Subnetz.

Eine weitere Möglichkeit ist die Verwendung von AWDS als Middleware. Die Anwendungen werden gegen die Middleware gelinkt und laufen im selben Prozesskontext. Der Vorteil dieses Ansatzes ist, dass die Anwendung Zugriff auf den internen Status des Routings hat. Damit lässt sich beispielsweise die Netzwerktopologie analysieren oder einzelne Attribute von AWDS-Paketen auswerten. Deshalb nutzen wir diese Variante auch in der vorliegenden Arbeit.

Eine dritte Variante ist die Verwendung von AWDS als Teil eines Simulationsmodells. Durch Verwendung einer Abstraktionsschicht [HM05] kann die Routing-Implementierung sowohl auf realer Hardware, als auch auf virtuellen Knoten im Netzwerksimulator NS-2[NS208] eingesetzt werden. Wir verwenden dies um effizient unterschiedliche Szenarien zu untersuchen.

4 P/S über WMNs

Wie bereits in Abbildung 1 dargestellt, setzt die COSMIC Middleware auf dem AWDS-Routing auf. Um einen besseren Zugriff auf interne Zustände des Routings zu erhalten, nutzen wir es als Middleware. Entsprechend laufen COSMIC, AWDS und die Anwendungen im selben Prozesskontext.

4.1 Verwaltung der Publisher/Subscriber

Für die Verwaltung der Publisher und Subscriber nutzen wir ein dezentrales Protokoll. Jeder Subscriber flutet periodisch die Liste seiner abonnierten Ereigniskanäle. Die entsprechenden Publisher empfangen diese Listen und vermerken den Absender, den Zeitpunkt und die Entfernung in Hops. Letztere wird aus dem Subscriber-Paket ermittelt, welches zusätzlich den initialen TTL-Wert enthält. Ein Empfänger kann aus der Differenz zum empfangenen Wert eine Entfernung abschätzen.

Ein erfasster Subscriber muss periodisch seine Subscription erneuern, ansonsten wird er bei den Publishern abgemeldet. Dieses Soft-State-Verfahren hat den Vorteil, dass ausgefallene oder vom Netz getrennte Subscriber erkannt und nicht mehr beachtet werden.

4.2 Effizientes Publizieren von Ereignissen

Das Publizieren von Ereignissen versucht möglichst effizient die Ressourcen des Netzwerks zu nutzen. Insbesondere sollen keine Ressourcen für das Publizieren von Ereignissen genutzt werden, wenn keine oder nur wenige Subscriber vorhanden sind. Im Allgemeinen werden Ereignisse im Netz geflutet. Die Pakete enthalten das Ereignis-Subjekt, Attribute und die Daten. Beim Empfänger filtert die COSMIC-Middleware die für sie relevanten Ereignisse heraus und reicht sie an die Anwendung weiter. Die Kosten hierfür sind relativ hoch, in einem Netz mit n Stationen wird die Nachricht (n-1)-mal wiederholt. Daher werden Optimierungen anhand der Liste der Empfänger vorgenommen.

Ist die Empfängermenge leer, so werden die Ereignisse bereits beim Publisher durch die Middleware verworfen, sie werden somit überhaupt nicht gesendet und damit die Bandbreite nicht unnötig beansprucht. Sind nur eine kleine Anzahl ($\leq k$) an Empfängern registriert, so werden die Daten per Unicast adressiert.

4.3 Reduktion der Flut-Reichweite

Häufig besitzen Publisher und Subscriber eine topologisch geringe Distanz zueinander. In solchen Fällen ist ein Fluten durch das gesamte Netzwerk nicht nötig und ließe sich durch Beschränkung der Reichweite optimieren. Durch die Entfernungsschätzung aus den Subscribe-Nachrichten kann die maximale Reichweite aller Empfänger bestimmt und diese als TTL beim Fluten verwendet werden.

Im Beispiel in Abbildung 2, wo der Publisher 3 zu den Subscribern 2 und 4 sendet, ist es ausreichend, eine TTL von 1 zu verwenden. Die Nachricht wird damit nur einmal gesendet und nicht 5-mal wiederholt.

Dieser theoretisch gute Ansatz hat sich bei praktischen Experimenten allerdings als suboptimal herausgestellt. Der zu beobachtende Effekt ist, dass die Reichweite oftmals unterschätzt wurde, wodurch die Flut-Pakete nicht alle Empfänger erreichen. Die Ursache dafür liegt in der Propagation von Broadcast-Paketen. Diese folgen nicht immer der Topologie, sondern können oftmals von Stationen mit mehreren Hops Entfernung empfangen werden. Sie „kürzen ab",

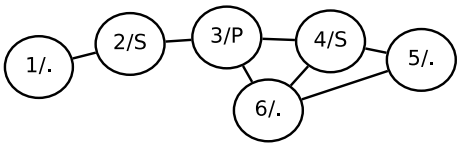


Abbildung 2. Beispiel für reduziertes Fluten

was in einer zu geringen Entfernungsschätzung resultiert. Hier ist Potential für weitere Arbeiten die zu robusteren Lösungen führen.

4.4 Aufwandsbetrachtung

Die Kommunikationkosten lassen sich mit der Anzahl der benötigten Sendevorgänge beschreiben. Beim Fluten in einer Topologie mit n Stationen werden genau n Sendevorgänge benötigt. Bei einer Unicast-Kommunikation entlang eines Pfades mit q Hops sind es entsprechend q. Die Paketgröße und der Einfluss der Senderate sind für diese Betrachtung zu vernachlässigen.

Wir unterscheiden die Verwaltung der Subscriber und die Propagation der Ereignisse. Erstere benötigen je n Nachrichten pro Subscriber, wobei dies nur selten, üblicherweise alle 60 Sekunden, geschieht. In typischen Topologien mit ca. 100 Stationen ist dieser Overhead immer noch vernachlässigbar gering. Bei der Publikation von Ereignissen ist die Anzahl der Subscriber relevant. Ist diese kleiner als k (vgl. Abschnitt 4.2), so werden einzelne Unicast-Routen verwendet. Bei nur einem Subscriber pro Publisher ist die Anzahl der Sendevorgänge offensichtlich minimal. Bei mehreren kann es vorkommen, dass die selben Nachrichten mehrfach (durch mehrere Unicast-Sendevorgänge) auf einem gemeinsamen Link übertragen werden. Durch Multicast-Bäume ist dies noch vermeidbar. Das entsprechende *K-minimum spanning tree problem* zurückzuführen auf das Steinerbaumproblem [PS02] ist allerdings ein bekanntes NP-vollständiges Problem und erfordert geeignete Heuristiken. Das Fluten hingegen benötigt offensichtlich die maximale Anzahl an Sendevorgängen, stellt gegenüber den anderen Mechanismen aber eine deutlich robusteres Verfahren dar.

Eine weitere Möglichkeit ist die Einführung einer zentralen Komponente (Broker), die sowohl die Subscriber verwaltet als auch die Ereignisse verbreitet. Hierdurch führt man aber einen Single-Point-of-Failure ein und der Broker-Knoten sowie dessen unmittelbare Umgebung würde einen Hotspot (Ort mit extremen Datenaufkommen) im Netz bilden. Der dezentrale Ansatz vermeidet dies, was zu einer homogeneren Verteilung der Last führt.

4.5 Gateway

Gateways in der COSMIC-Architektur dienen der Vermittlung zwischen mehreren Netzen. In der Abbildung 1 stellt ein COSMIC-Gateway eine Verbindung zwischen einem COSMIC-AWDS-Netz und einem COSMIC-Feldbus (z.B. CAN) her.

Die Hauptaufgaben, die ein Gateway im Allgemeinen zu erfüllen hat, sind vornehmlich Filtern, Formattransformationen und gegebenenfalls Attributanpassungen. Filtern bedeutet, Ereignisse nur dann in ein anderes Netzwerk zu übertragen, wenn sie in dem entsprechenden Teilnetz abonniert sind oder sie andernfalls zu verwerfen. Neben der allgemeinen Filterung kann zusätzlich eine Filterung etwa nach [KM03] auf den Attributen der Ereignisse durchgeführt werden, wenn diese bspw. eine Weiterleitung über Netzwerkgrenzen verbieten oder die Gültigkeit eines Ereignisses abgelaufen ist.

Neben den Filtereigenschaften muss das Gateway Anpassungen an weiterzuleitenden Ereignissen durchführen. Die Transformation eines Ereignisses in das Format des anderen Netzes ist deshalb von Bedeutung, als dass in den verschiedenen Netzwerken auf der untersten Ebene unterschiedliche Nachrichten ausgetauscht werden bzw. die Art der Adressierung unterschiedlich ist. Zusätzlich zum Format müssen gegebenenfalls die Attribute des Ereignisses adaptiert (z.B. Änderung der Güteeigenschaften) bzw. mit Informationen (z.B. Ort) angereichert werden.

Zusammenfassend werden Ereignisse folglich nur selektiv von einem Netz in das andere übertragen und somit garantiert die Gatewayfunktionalität gleichzeitig den sparsamen Umgang mit der zur Verfügung stehenden Bandbreite.

5 Evaluierung

Für die Evaluierung wurde ein einfaches aber hinreichend repräsentatives Szenario gewählt. Wir verwenden 9 Stationen, die eine ringförmige Topologie bilden (vgl. Abbildung 3). Um diese zu generieren wurden Paketfilter zwischen den Stationen konfiguriert, die verhindern dass sich andere als die vorgegebenen Verbindungen etablieren. Als Stationen dienen Linux-Workstations mit 2.4 GHz Prozessoren und Atheros-basierten WLAN-Netzwerkkarten Die Rechner sind über NTP mit einer Genauigkeit von ca. 1 ms synchronisiert.

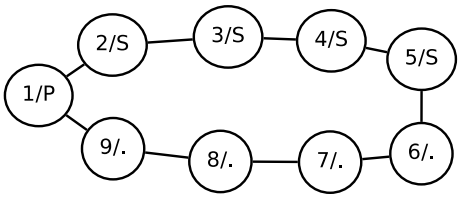


Abbildung 3. Beispiel-Topologie

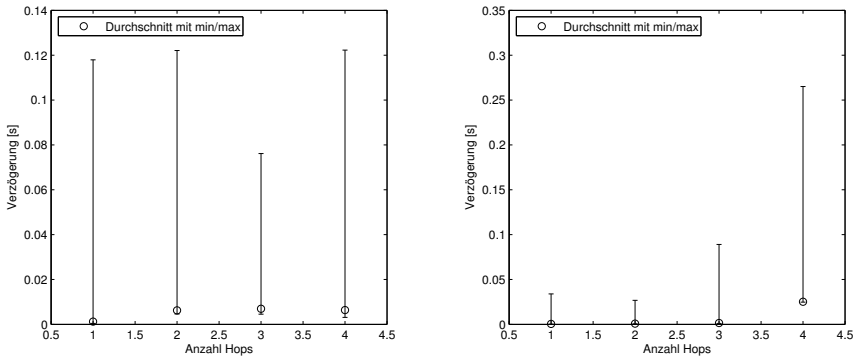
Die Publisher-Anwendung läuft auf Station 1, die Subscriber auf den Stationen 2–5. Hierdurch können 1–4 Hops untersucht werden. Als Daten werden 32-Bit-Sequenznummern mit einer Periode von 1,5 s übertragen, die eine einfache Analyse des Paketverlustes erlauben. Sowohl auf Publisher- als auch auf

Subscriber-Seite werde Sende- und Empfangs-Zeitstempel auf Anwendungsebene protokolliert und anschließend die Zeitdifferenzen berechnet. Die folgenden Ergebnisse stellen jeweils Minimum, Maximum und Arithmetisches Mittel von 1500 Ereignissen dar.

Betrachtet werden zwei Varianten. Bei der ersten sind alle Subscriber gleichzeitig aktiv, wodurch die Broadcast-Kommunikation verwendet wird. Bei der zweiten ist jeweils nur ein Subscriber aktiv, weshalb Unicast-Pakete genutzt werden. Auf diese Weise lassen sich die Unterschiede zwischen beiden Verfahren aufzeigen.

5.1 Messungen

Die Ende-zu-Ende-Verzögerungen unter Verwendung des Flutens sind in Abbildung 4(a) und die für Unicast-Übertragungen in Abbildung 4(b) dargestellt. Die durchschnittlichen Werte zeigen den zu erwartenden Anstieg mit zunehmender Hop-Anzahl. Weiterhin ist zu erkennen, dass Übertragungszeiten von weniger als 300 ms erreicht werden, was für sehr viele Anwendungen geeignet ist.



(a) Broadcast (b) Unicast

Abbildung 4. Messungen, Paket-Verzögerung pro Hops

Der durchschnittliche Paketverlust lag beim Broadcast-Szenario bei ca. 3 %. Dies ist ein relativ hoher Wert, der aus der hier verwendeten Ringtopologie resultiert. Normalerweise hat eine Station mehr als nur zwei Nachbarn. Damit ist auch die Wahrscheinlichkeit höher, ein Flut-Paket über andere Wege zu empfangen. Aus anderen Experimenten wissen wir, dass bei stärker vermaschten Topologien die Verlustrate wesentlich geringer ist, meist unter 1 %.

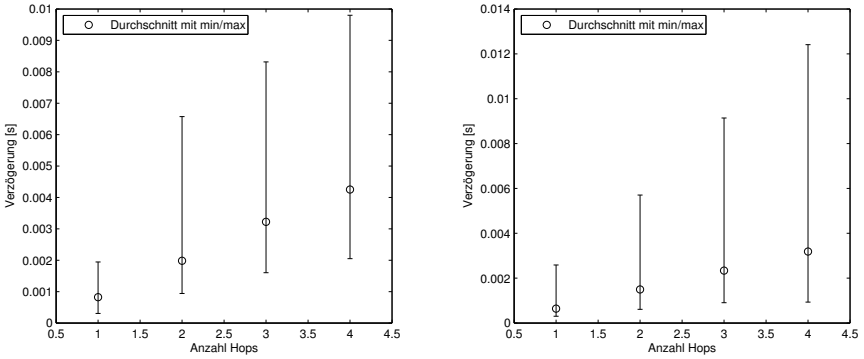
Beim Unicast-Szenario lag der durchschnittliche Paketverlust unter 0,1 %. Dies resultiert aus dem selbstständigen erneuten Übertragen von verlorenen Paketen durch den WLAN-MAC-Layer.

Die zweite Messung stellt die Ende-zu-Ende-Verzögerung bei Verwendung von Unicast-Paketen dar. Die Ergebnisse sind vergleichbar mit den vorherigen. Die maximalen Verzögerungen heben sich stark vom Durchschnitt ab, was auf einzelne Ausreißer hinweist. Diese könnten z.B. an Verzögerungen im Betriebsystem (Prozess-Scheduling) oder dem WLAN-Treiber liegen. Eine weitere Fehlerquelle könnte die Uhrensynchronisation zwischen den Rechnern sein.

5.2 Simulation

Durch die Granularität der Uhrensynchronisation von nur 1 ms weisen die Messungen eine schlechte Auflösung auf. Diese können in der Simulation vermieden werden. Wir verwenden einen um die Abstraktionsschicht (vgl. Abschnitt 3) erweiterte Version des Netzwerksimulators NS-2 (Version 2.32). Das Szenario wurde identisch im Simulator nachgestellt. Die Datenrate wurde auf 2 MBit/s eingestellt. Die Routing-Software, Middleware und Testanwendungen sind identisch mit denen aus der realen Messung.

Die resultierenden Werte sind in Abbildung 5(a) und 5(b) dargestellt. In der Simulation treten nicht die Probleme der Uhrensynchronisation auf. Allerdings sind die Übertragungszeiten auch wesentlich geringer. Wie bereits in [HLI06] untersucht, scheint dies an dem Nicht-Berücksichtigen der Verzögerungen in der Laufzeitumgebung zu liegen.



(a) Broadcast (b) Unicast

Abbildung 5. Simulation, Paket-Verzögerung pro Hops

Wie zu erwarten war, zeigt sich hier deutlich die lineare Abhängigkeit der Latenz von der Anzahl der Hops. Die Unicast-Übertragung hat etwas geringere Übertragungszeiten. Sie liegen aber in der selben Größenordnung wie die Broadcast-Übertragungen. Die maximalen Zeiten sind wesentlich geringer als bei realen Messungen. Dies deutet darauf hin, dass Ausreißer tatsächlich durch Jitter in der Laufzeitumgebung verursacht werden.

In der Simulation traten keine Paket-Verluste auf, was unter anderem an dem idealisierten Modell der Funkübertragung liegt.

6 Verwandte Arbeiten

Aus der Literatur sind eine Reihe unterschiedlicher Publish/Subscribe Systeme bekannt, welche auf unterschiedliche Gebiete fokussieren bzw. Voraussetzung an die zugrundeliegende Infrastruktur stellen. Arbeiten wie SIENA [CRW01], READY [GKP99] oder HERMES [PB02] bieten ein skalierbares P/S-System, setzen aber ein statisches verteiltes Ereignis-Broker-Servernetzwerk mit zuverlässige TCP-Kommunikation voraus. Die Voraussetzungen und die damit verbundene Schwergewichtigkeit schließt jedoch den Einsatz auf Mikrocontrollern aus. Darüberhinaus sind diese Systeme nicht selbstorganisierend, sondern erfordern für den Aufbau der Backbone-Server administrativen Einsatz.

Im Bereich der Mobilen Ad-Hoc-Netzwerke wurden spezifische Lösungen entwickelt, wie beispielsweise STEAM [MC02] oder der Ansatz von Mahrenholz [Mah06]. Diese betrachten jedoch eher Fragestellungen der Mobilität und der effizienten Strukturierung von Datenströmen.

Im Bereich der Sensor/Aktor-Systeme bzw. Wireless Sensor Networks (WSN) gibt es ebenfalls einige Systeme, die die Heterogenität der unterliegenden Hardware bzw. Kommunikationstechnologie verbergen und dabei unterschiedliche Ziele verfolgen. Zumeist ist die Kommunikation im WSNs von den Knoten in Richtung einer einzigen Senke organisiert (bspw. [SGGV+04]). Weiterhin liegt das Hauptaugenmerk in WSNs auf dem sparsamen Umgang mit der Ressource Energie, welches in unserem Einsatzgebiet nicht relevant ist.

Es gibt folglich eine große Anzahl unterschiedlicher Publish/Subscribe-Systeme, welche jedoch meist auf relativ homogener Hardware und standardisierter Netzwerktechnologie aufsetzen. Unser hier vorgestellte System aus der Verbindung zwischen COSMIC und AWDS erlaubt jedoch den Einsatz über ein breites Hardwarespektrum angefangen von Mikrocontrollern bis hinzu PC unter Beibehaltung der gleichen Kommunikationsschnittstelle für die Anwendung. Die Übergänge zwischen den Systemen und Netzwerken ist auf Anwendungsebene vollständig transparent.

7 Fazit und Ausblick

In dieser Arbeit präsentieren wir einen aktuellen Forschungsansatz für den Einsatz von Publish/Subscribe-Kommunikation in Wireless Mesh Networks. Durch den Einsatz der COSMIC Middleware wird die Kommunikation auf extrem heterogene Systeme ausgedehnt, bis hin zu 8-Bit-Mikrocontrollern an Feldbussen. Die Kommunikation wird dabei dynamisch durch die Middleware selbst organisiert. Es zeigt sich hierdurch auch die Flexibilität der Middleware, die mit geringem Aufwand WMNs als Transportschicht integriert.

Die Paketverluste und Übertragungszeiten aus Messungen und Simulationen zeigen, dass ein solches System sehr gut für viele Anwendungen geeignet

ist. Allerdings können bisher nur Best-Effort-Daten übertragen werden. Für das AWDS-Routing wurden schon einige Erweiterungen entwickelt, die Reservierungen von Bandbreiten erlauben [HLI07, HSL07]. Diese sind bisher aber noch nicht in die Middleware integriert. Hierauf wird sich die weitere Arbeit konzentrieren, wodurch dann auch Soft-Realtime-Kommunikation ermöglicht wird.

Literatur

[AWD08] AWDS project homepage, 2008. online, http://awds.berlios.de.

[CRW01] Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Trans. Comput. Syst.*, 19(3):332–383, 2001.

[GKP99] R. Gruber, B. Krishnamurthy, and E. Panagos. The architecture of the ready event notification service, 1999.

[HLI06] André Herms, Georg Lukas, and Svilen Ivanov. Realism in design and evaluation of wireless routing protocols. In Otto Spaniol, editor, *Proceedings of First international Workshop on Mobile Services and Personalized Environments (MSPE'06)*, volume P-102, pages 57–70, Aachen, Germany, November 2006. Lecture Notes in Informatics (LNI).

[HLI07] André Herms, Georg Lukas, and Svilen Ivanov. Measurement-based detection of interfering neighbors for qos in wireless mesh networks. In *16th IST Mobile and Wireless Communications Summit 2007, Proceedings of*, 2007.

[HM05] André Herms and Daniel Mahrenholz. Unified development and deployment of network protocols. In *Proceedings of MeshNets Workshop in conjunction with First International Conference on Wireless Internet (WICON)*, Budapest, Hungary, July 2005.

[HSL07] André Herms, Stefan Schemmer, and Georg Lukas. Real-time mesh networks for industrial automation. In *Proceedings of SPS/IPC/DRIVES, Elektrische Automatisierung, Systeme und Komponenten '07*, Nurenberg, Germany, November 2007.

[KB02] J. Kaiser and 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, August 28–30 2002.

[KBMP03] J. Kaiser, C. Brudna, C. Mitidieri, and C. Pereira. COSMIC: A middleware for event-based interaction on CAN. In *9th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2003)*, Lisbon, Portugal, September 2003.

[KM03] Jörg Kaiser and Carlos Mitidieri. Attribute-based filtering for embedded systems. In *Proceedings of the 2nd International workshop on Distributed event-based systems*, pages 1– 7, San Diego, California, 2003.

[Mah06] Daniel Mahrenholz. *Providing QoS for Publish/Subscribe Communication in Dynamic Ad-hoc Networks.* Doctoral dissertation, University of Magdeburg, May 2006. online http://diglib.uni-magdeburg.de/Dissertationen/2006/danmahrenholz.pdf.

[MC02] René Meier and Vinny Cahill. STEAM: Event-Based Middleware for Wireless Ad Hoc Network. In *ICDCSW '02: Proceedings of the 22nd International Conference on Distributed Computing Systems*, pages 639–644, Washington, DC, USA, 2002. IEEE Computer Society.

[NS208] NS-2 project homepage, 2008. online, http://nsnam.sf.net.

[PB02] Peter R. Pietzuch and Jean Bacon. Hermes: A distributed event-based middleware architecture. In *ICDCSW '02: Proceedings of the 22nd International Conference on Distributed Computing Systems*, pages 611–618, Washington, DC, USA, 2002. IEEE Computer Society.

[PS02] Hans Jürgen Prömel and Angelika Steger. *The Steiner Tree Problem: A Tour Through Graphs, Algorithms and Complexity.* Vieweg-Verlag, Feb 2002.

[Rob91] Robert Bosch GmbH. *CAN Specification Version 2.0.* 1991.

[SGGV+04] Eduardo Souto, aes Germano Guimar Glauco Vasconcelos, Mardoqueu Vieira, Nelson Rosa, and Carlos Ferraz. A message-oriented middleware for sensor networks. In *MPAC '04: Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing*, pages 127–134, New York, NY, USA, 2004. ACM.

[SZ07] Michael Schulze and Sebastian 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, September 11–12 2007.

[Zig03] ZigBee Alliance. *ZigBee Specification - IEEE 802.15.4.* 2003.