Ein maßgeschneidertes Kommunikationssystem für eine mobile Applikation mit Dienstgüteanforderungen

C.Webel, I.Fliege und A.Geraldy

{webel, fliege, geraldy}@informatik.uni-kl.de
AG Vernetzte Systeme, TU Kaiserslautern
Kaiserslautern, Deutschland

Technischer Bericht 336/04

Fachbereich Informatik
Technische Universität Laiserslautern
Postfach 3049
67653 Kaiserslautern
Deutschland

Ein maßgeschneidertes Kommunikationssystem für eine mobile Applikation mit Dienstgüteanforderungen

Christian Webel, Ingmar Fliege und Alexander Geraldy

{webel, fliege, geraldy}@informatik.uni-kl.de
AG Vernetzte Systeme, TU Kaiserslautern
Kaiserslautern, Deutschland

Zusammenfassung In diesem Beitrag wird die Maßschneiderung eines Ad-Hoc-Kommunikationssystems zur Fernsteuerung eines Luftschiffs über WLAN vorgestellt. Dabei steht die Dienstunterstützung bei der Übertragung mehrerer Datenströme im Vordergrund. Es werden verschiedene Dienstgütemechanismen erklärt und deren Entwicklung und Integration in ein Kommunikationsprotokoll mit Hilfe eines komponentenbasierten Ansatzes genauer erläutert.

1 Einleitung

Die Entwicklung von Kommunikationsprotokollen für drahtlose mobile Ad-Hoc-Kommunikationsnetze stellt eine besondere Herausforderung dar. Diese Ad-Hoc-Netze bestehen aus einer Menge von Knoten, die über eine Funkschnittstelle kommunizieren und sich dabei selbst organisieren. Gerade diese Selbstorganisation gestaltet sich durch die fehlende Infrastruktur und die Mobilität von Knoten als besonders schwierig, da sich die Netzwerktopologie nicht zuverlässig erfassen lässt und die Verbindungen zwischen den Knoten ständigen Veränderungen und Störungen unterworfen sind. Aus dem gleichen Grund sind anwendungsspezifische Anforderungen an die Dienstgüte (z.B. die Performanz) und die Zuverlässigkeit der Kommunikation nur schwer zu erfüllen.
Die eingeschränkten Ressourcen, die häufig in mobilen Systemen vorzufinden sind, erfordern zusätzlich schlanke, maßgeschneiderte Kommunikationsprotokolle, die auf die speziellen Anforderungen einer Anwendung zugeschnitten sind. Dabei spielt Dienstgüte eine wichtige Rolle, da sie in vielen Teilen (Medienzugriff, Routing, Flusskontrolle) eines Kommunikationsprotokolls betrachtet werden muss.
Noch heute bildet das in den 80er Jahren eingeführte OSI-Schichtenmodell eine wichtige Grundlage in der Kommunikationsbranche. Verwendung findet es vorwiegend bei der Beschreibung und der Gruppierung einzelner Funktionalitäten. Große Teile von Kommunikationsprotokollen werden schichtenübergreifend, monolithisch implementiert (vgl. Anwendungen im WWW). Um aber die Komplexität zu beherrschen und die Wiederverwendung von Softwareartefakten zu ermöglichen, ist eine feingranulare Strukturierung notwendig. Das durch das OSI-Schichtenmodell

gegebene Framework ist hierfür ungeeignet, da es keine ausreichende Flexibilität bietet [1]. Der in dieser Anwendung verwendete Mikroprotokollansatz [1,2] dagegen ermöglicht durch eine komponentenbasierte Protokollentwicklung eine sehr feine Strukturierung und flexible Architekturen, wodurch Maßschneiderung und Wiederverwendung unterstützt werden.
Die hier vorgestellte Kommunikationsmiddleware mit Dienstgüteunterstützung wurde zur Fernsteuerung eines Fluggerätes mit Video- und Datenübertragung maßgeschneidert (siehe Abb. 1). Ausgehend von einer Dienstgütespezifikation [3] wird mit Hilfe verschiedener Dienstgütemechanismen [3] die Behandlung von unterschiedlichen, periodisch auftretenden Daten vorgenommen.

Abbildung 1. Vorführung des Luftschiffes

Dieser Beitrag ist wie folgt gegliedert. In Kapitel 2 wird die Anwendung samt Anforderungen [4] vorgestellt, für die eine maßgeschneiderte Kommunikationsmiddleware in Kapitel 3 beschrieben wird. Ausgewählte Protokollkomponenten, die bei der Entwicklung verwendet wurden, werden genauer erläutert. Kapitel 4 fasst die Ergebnisse zusammmen und gibt einen kurzen Ausblick.

2 Anwendung „Luftschiff“

Die Anwendung besteht aus zwei Teilsystemen, einer Videoanwendung und einer Steuerungsanwendung. Ein Videoserver im Luftschiff überträgt in periodischen Abständen Bilder, die von einem oder mehreren Videoclients entgegengenommen werden. Der Steuerungsserver, die Steuerungs-

komponente des Luftschiffes, nimmt von einem Steuerungsclient Kommandos entgegen und regelt die Motoren (Heck und Auftrieb) und den Servo (horizontale Ausrichtung der Auftriebsmotoren) ein. Die Anwendungen sind verteilt und kommunizieren über WLAN miteinander. Eine vollständige Beschreibung der Anwendung ist in [4] zu finden.

2.1 Die Videoanwendung

Der Client der Videoanwendung verarbeitet die Benutzereingaben und stellt die Videodaten und -parameter des Luftschiffes geeignet dar. Die Funktionalitäten der Videoanwendung lassen sich in zwei Gruppen unterteilen, die Sende- bzw. Empfangsfunktionalitäten und die Dienstgütefunktionalitäten. Die Sende- und Empfangsfunktionalitäten umfassen folgende Punkte:

- Senden der Bilder (Server) über WLAN
- Empfangen/Anzeigen der Bilder (Client)

Die Dienstgütefunktionalitäten sind:

- Hinzufügen und Ändern von Dienstgüteklassen (Client)
- Setzen der aktuellen Dienstgüteklasse (Server)

Dienstgüteklassen beschreiben mögliche Anwendungsszenarien, die verschiedene Anforderungen (Auflösung/Qualität und Bildrate) an die Übertragung haben:

- Suchflug (search): Erfordert hohe Bildqualität zur Personensuche, bei mittlerer bis hoher Bildrate.
- Überwachungsflug (watch): Mittlere bis hohe Bildqualität bei hoher bis sehr hoher Bildrate zur Erkennung von kleineren Bewegungen.
- Panoramaflug (panorama): Sehr hohe Bildqualität, Bildrate nebensächlich.
- Soloflug (solo): Es findet keine Bildübertragung statt, da das Luftschiff aus unmittelbarer Umgebung gesteuert wird.

2.2 Die Steuerungsanwendung

Der Server im Luftschiff empfängt die Steuerkommandos und kontrolliert das Fluggerät. Der aktive Steuerungsclient nimmt Benutzereingaben in Form von Steuerkommandos und Dienstgüteklassen entgegen, sendet diese zum Server und stellt die aktuellen Luftschiffparameter graphisch dar. Die Steuerung bietet dazu folgende Funktionalitäten:

- Steuerung des Luftschiffes (inklusive Senden und Empfangen der Steuerkommandos)
- Übertragen und Anzeigen der aktuellen Luftschiffparameter (aktuelle Steuerdaten, Restkapazität des Akkus).
- Hinzufügen und Ändern von Dienstgüteklassen (Client), Setzen der aktuellen Dienstgüteklasse

Analog zur Videoanwendung liegen auch hier verschiedene Anforderungen an die Übertragung der Steuerkommandos vor, die je nach Szenario variieren können: Eine günstige Flugumgebung (easy) und eine ungünstige Flugumgebung (hard). Bei letzterer wird eine sehr kurze Verzögerung der Steuerkommandos gefordert.

Systemebene	geforderte Dienstgüte	gewährte Dienstgüte
Benutzer	search, watch, panorama, solo	Grün, Gelb, Rot
Anwendung	Bilder/Sekunde, Qualität	Bilder/Sekunde, Qualität
Middleware	MwDgKlasse	Datenvolumen, Periode
Basisdienst	#Zeitschlitze/Sekunde	#Zeitschlitze/Sekunde

Tabelle 1. Dienstgüteparameter auf verschiedenen Schichten (Video)

MwDgKlasse	Parameter	Beschreibung
konstant	const	konstantes *const* Datenvolumen und Periode
variabel	min	minimales *min* Datenvolumen und Periode
	max	gewünschtes *max* Datenvolumen und Periode

Tabelle 2. Dienstgüteklassen auf Middleware-Ebene (MwDgKlasse)

2.3 Dienstgütespezifikation der Anwendung

Alle Dienstgütemechanismen sind auf eine geeignete Spezifikation angewiesen. Dabei sind zwei Punkte zu beschreiben: Zum einen die Dienstgüte, die vom Benutzer gefordert wird. Auf dieser Ebene sind dies die beschriebenen Anwendungsszenarien. Zum anderen die Dienstgüte, die bereitgestellt wird. Abhängig von den verfügbaren Ressourcen wird z.B. dem Benutzer ein entsprechendes Feedback gegeben (Grün: akzeptabel, Gelb: mininmal, Rot: unbrauchbar). Die verschiedenen Schichten des Kommunikationssystems benutzen jeweils einen anderen Abstraktionsgrad und daher auch eine andere Dienstgütespezifikation. Die Umsetzung zwischen diesen Spezifikationen erfolgt durch das *Dienstgüte-Mapping* (Kapitel 3.1). Tabelle 1 fasst die Dienstgüteparameter der Videoanwendung zusammen, die auf den verschiedenen Ebenen des Systems benutzt werden. Die Dienstgüteklassen *MwDgKlasse* der Middleware lassen sich in Tabelle 2 ablesen.

Performanz: Die Spezifikation der *Performanz* beschreibt die benötigten Kommunikationsressourcen einer Anwendung, z.B. Durchsatz, Delay, Jitter. Bei der Steuerung eines Fluggerätes spielt die maximale Verzögerung der Steuerkommandos eine große Rolle, dagegen steht bei der Videoübertragung der Durchsatz im Vordergrund.
Die endgültige *Performanz* bestimmt sich in dieser Anwendung erst durch die Definition des *Dienstgüte-Mappings*, da dieses Mapping die Abbildung der Dienstgüteklassen auf konkrete Performanzwerte (Delay, Jitter, Bandbreite, ...) festlegt.

Verbindlichkeit: Die *Verbindlichkeit* beschreibt die qualitative Festlegung der Dienstgüte. Man unterscheidet zwischen *deterministisch*, *statistisch* und *best effort* [5]. In mobilen Ad-hoc Netzen ist es bestenfalls möglich, *statistische* Garantien zu geben. Dazu müssen Reservierungen auf dem Medium getätigt werden können. Tabelle 3 zeigt die Festlegung

Anwendung	Verbindlichkeit	Priorität
Steuerung	statistisch	1
Videoübertragung	statistisch	2
Luftschiffparameter	statistisch	3

Tabelle 3. Spezifikation der Verbindlichkeiten

der Prioritäten der vorliegenden Anwendungen. Im Falle eines Engpasses erhält die Anwendung mit höchster Priorität (hier: 1, die Steuerung) die angeforderte Dienstgüte, während die Videoübertragung und die Luftschiffparameterübertragung dann ausgesetzt werden können.

Dienstgüte-Management-Richtlinien: Im Falle einer Dienstgüteverletzung, also einer Nichterfüllung oder Nichteinhaltung der reservierten Dienstgüte, müssen geeignete Maßnahmen ergriffen werden. Diese werden durch das *Dienstgüte-Management-Richtlinien* beschrieben. Die Übertragung der Videodaten stellt in diesem Fall die schwierigere Anwendung dar. Kann die reservierte Bandbreite vom Basisdienst nicht aufrecht gehalten werden, so müssen bei der Videoübertragung die Bildrate und die Qualität der Bilder und bei der Steuerung die Periode an die neue Situation angepasst werden. Die vollständigen Spezifikationen der Teilanwendungen finden sich in [4].

3 Maßschneiderung des Entwurfs

3.1 Dienstgüte in der Middleware

Dienstgüte-Bereitstellung: Sie beschreibt die Umsetzung der Spezifikationen in eine Ressourcenbelegung/-verteilung, damit Kommunikation in der vereinbarten Dienstgüte stattfinden kann [3]. Wichtige Punkte hierbei sind das *Dienstgüte-Mapping*, der *Zugangstest* und das *Reservierungsprotokoll*. Wir beschränken uns hier auf das Dienstgüte-Mapping, das die Abbildung zwischen Dienstgütespezifikationen verschiedener Abstraktionsebenen festlegt. Tabelle 4 zeigt exemplarisch das Dienstgüte-Mapping für die Videoanwendung. Den vier identifizierten Dienstgüteklassen auf Benutzerebene wird je eine minimale und eine gewünschte Dienstgüte zugewiesen, die speziell auf die Anforderungen des Benutzers angepasst sind (vgl. Tabelle 1). Die Middleware überführt die gegebene Bildrate und Bildgröße (Qualität) in ein Datenvolumen und eine feste Periode. Dazu wird die Middleware-Dienstgüteklasse *variabel* gewählt, die eine Festlegung des minimalen und gewünschten Datenvolumen/Periode-Tupels ermöglicht (entsprechend der minimalen und gewünschten Dienstgüte auf Anwendungsebene). Tabelle 5 beschreibt das Mapping des Feedbacks der Videoanwendung von Systemebene auf die Benutzerebene, abhängig von der bereitgestellten Bildrate und -qualität. Analog dazu existiert auch ein Mapping für die Übertragung der Steuerkommandos und der Luftschiffparameter.

Benutzer gewählte Dienstgüte	**Anwendung** minimale Dienstgüte	**Middleware** minimale Dienstgüte
search	15,50[1]	840, 1000[2]
watch	15,25	640, 1000
panorama	5,75	400, 1000
solo	0,0	0, 0
	gewünschte Dienstgüte	gewünschte Dienstgüte
search	20,75	1600, 1000
watch	25,50	1400, 1000
panorama	10,90	1440, 1000
solo	0,0	0, 0

1 (Bilder/Sek.[1/s], Qualität [%]) JPEG-Qual. 100% ≙ keine Kompression

2 (Volumen [kBit], Periode [ms])

Tabelle 4. Mapping der Benutzeranforderungen (Video)

Middleware	**Anwendung**	**Benutzer**
bereitgestellte Dg = max	Grün	akzeptabel
min ≤ bereitgestellte Dg < max	Gelb	minimal
bereitgestellte Dg < min	Rot	unbrauchbar

Tabelle 5. Mapping des Feedbacks

Dienstgüte-Kontrolle: Die Dienstgütekontrolle befasst sich mit allen kurzfristigen Maßnahmen, die nötig sind, um Kommunikation in der geforderten Dienstgüte bereitzustellen [3]. Diese Aufgabe muss zum einen die maßgeschneiderte Middleware übernehmen, zum anderen auch die Basistechnologie. Je nachdem ob die Basistechnologie Reservierungen für einen Link oder für eine Anwendung tätigt, müssen andere Mikroprotokolle verwendet werden.

Dienstgüte-Management: Unter Dienstgüte-Management fasst man alle längerfristigen Maßnahmen zusammen, um Kommunikation mit vereinbarter Dienstgüte zu ermöglichen und zu überwachen [3]. Das genaue Management wird durch die *Management-Richtlinien* in der Dienstgüte-Spezifikation beschrieben und umfasst unter anderem folgende Punkte:

- Die Aufgabe des *Dienstgüteerfassung* (*monitoring*) ist die Überwachung und Erfassung der aktuellen Verfügbarkeit von Ressourcen und Dienstgüteparametern. Der momentane Zustand (Durchsatz, Delay, Jitter, Pufferauslastung) der jeweiligen Ende-zu-Ende Verbindung und des Netzes bildet die Grundlage für verkehrskontrollierende Funktionen. So können durch ständiges Überwachen Verbindungsausfälle oder andere Fehler rechtzeitig erkannt und behandelt werden.
- Die *Dienstgüteskalierung* (*scaling*) gliedert sich in Dienstgütefilterung und Dienstgüteanpassung. Unter Filterung versteht man

die Manipulation von Datenströmen oder Paketfolgen während der Übertragung, indem gezielt Daten verworfen werden. So können bestimmte Daten ausgeblendet werden, falls es zeitweise keine Geräte im Netz gibt, die diese Daten brauchen oder man zu bestimmten Zeiten nur eine Teilmenge der Daten benötigt. Die Adaption beschreibt die senderseitige Anpassung des Datenstroms, um auf eine geänderte Ressourcensituation auf dem Medium zu reagieren.

Im Gegensatz zu den Dienstgütekontrollmaßnahmen bezieht sich die obige Dienstgüteanpassung nicht auf eine Regulierung des vorhandenen Datenstromes, sondern auf eine senderseitige Anpassung des Datenstromes. Darunter fällt die Anpassung der Bildgröße und der Framerate einer Kamera, und auf eine veränderte Ressourcensituation zu reagieren oder die Vergrößerung der Periode der Steuerkommandos, um die Datenrate zu senken.

3.2 Die Architektur des Kommunikationssystems

Die Entwicklung von Kommunikationsprotokollen ist eine komplexe Aufgabe, da es sich um verteilte, nebenläufige Systeme mit Anforderungen an Synchronisation, Fehlerbehandlung und Dienstgüte handelt. Um diese Komplexität zu beherrschen, ist ein methodisches Vorgehen und die Verwendung von formalen Beschreibungssprachen wie SDL [6] wesentlich.
Das hier vorgestellt Kommunikationssystem wurde mit dem Mikroprotokollansatz entwickelt [2]. Ein Mikroprotokoll ist eine abgeschlossene, sofort einsetzbare Komponente mit wohldefinierten Schnittstellen, die eine einzelne Protokollfunktionalität kapselt. Mikroprotokolle werden aus einer Mikroprotokollbibliothek selektiert und zu einem vollständigen, maßgeschneiderten Kommunikationssystem komponiert.
Abbildung 2 zeigt die Struktur und Architektur des Systems. Die Luftschiffapplikation (Server) ist in einen Videoserver und einen Steuerungsserver unterteilt. Diese kommunizieren über eine Middleware mit den entsprechenden Hardwarekomponenten (Kamera, Servo/Motoren) und den Basisstationen (Client). Da wir ein maßgeschneidertes Kommunikationssystem entwickeln, werden client- und serverseitig dedizierte Middlewares eingesetzt, die auf die jeweiligen Bedürfnisse der darauf aufsetzenden Applikation zugeschnitten sind. Abbildung 3 zeigt die maßgeschneiderte Architektur der Luftschiff-Middleware. Die Middleware ist zweigeteilt: Zum einen in eine anwendungsspezifische *(VideoServerAppMw* und *PilotServerAppMw)* und zum anderen in eine dienstgütespezifische Middleware (*QosMw*). Die jeweiligen Middlewareblöcke kommunizieren über feste Schnittstellen mit der Applikation, der Basistechnologie oder untereinander. Analog zur Struktur der Middleware auf der Serverseite ergibt sich die Struktur auf Clientseite.

3.3 Die applikationsspezifische Middleware

Es gibt insgesamt vier verschiedene applikationsspezifische Middlewares, die auf die jeweiligen Teilanwendungen des Luftschiffs und der Basis-

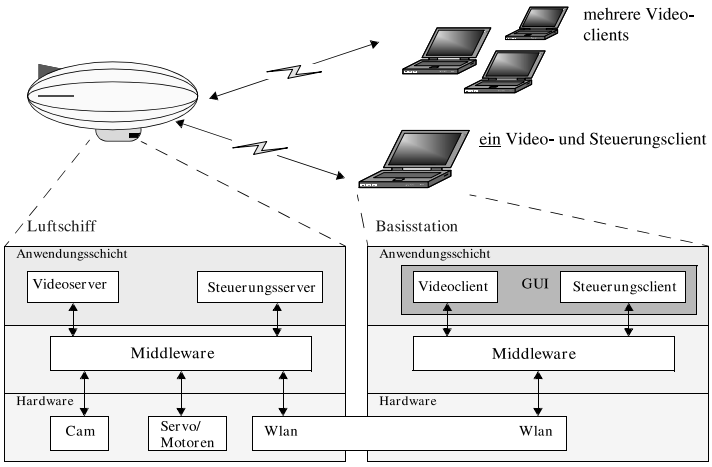

Abbildung 2. Struktur und Architektur des Kommunikationssystems

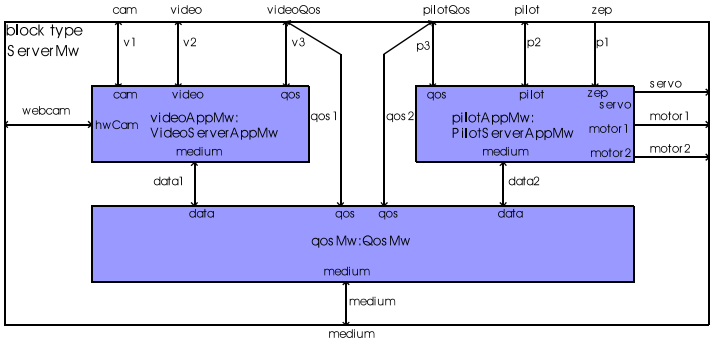

Abbildung 3. Struktur der Middleware (Server)

stationen maßgeschneidert sind. Der Vorteil einer solcher maßgeschneiderten Middleware ist der geringere Ressourcenbedarf, da sie nur die minimal benötigten Funktionalitäten bereitstellt. Abbildung 4 zeigt die Struktur der applikationsspezifischen Middleware der Videoübertragung des Luftschiffes.
Die Komponente *CamMgr* dient zur Kontrolle der installierten Kamera, *VideoCoDec* sorgt für eine Transparenz der Middleware aus Anwendungssicht, indem es die verschiedenen Signale zwischen der Client- und Serveranwendung en- bzw. decodiert und serialisiert. Die beiden Dienstgütemechanismen *Dienstgütemapping* und *Dienstgüteskalierung* wurden jeweils im Kontext der Videoübertragung als Mikroprotokolle spezifiziert:

VideoMapping: Das Mikroprotokoll *VideoMapping* dient der applikationsabhängigen Umsetzung der Dienstgütespezifikation zwischen der Anwendungs- und Systemebene (Middleware). Es ermöglicht die Defini-

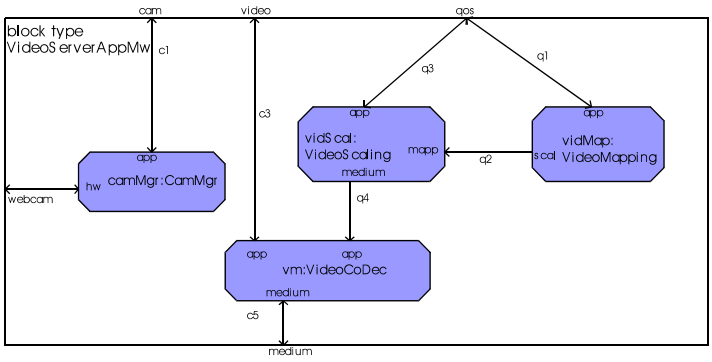

Abbildung 4. Struktur der appl.-spez. Middleware (Luftschiff)

tion neuer und die Auswahl vorhandener Dienstgüteklassen. Nach der Auswahl einer Dienstgüteklasse werden die notwendigen Dienstgüteparameter an die Anwendung (zur Steuerung der Kamera) und an die Skalierungskomponente *VideoScaling* übermittelt. Es realisiert somit die Dienstgüte-Bereitstellung.

VideoScaling: Das Mikroprotokoll *VideoScaling* übernimmt die Dienstgüteanpassung als Teil des *Dienstgüte-Managements*. Es verarbeitet die Bilder der Kamera und Information über die aktuelle Ressourcensituation, um die Parameter der Kamera an die aktuelle Verkehrssituation anzupassen. Die Anpassung erfolgt sowohl über die Bildrate als auch über die JPEG-Bildqualität. Da diese Komponente die aktuelle Ressourcensituation und die gewählte Dienstgüteklasse der Anwendung kennt, kann sie dem Benutzer ein entsprechendes Feedback nach Tabelle 5 geben. Eine Filterung von Daten findet bei der Videoübertragung nicht statt.

3.4 Die applikationsunabhängige Middleware

Die applikationsunabhängige Middleware (siehe Abb. 5) realisiert die generischen Dienstgütefunktionalitäten. Sie baut auf einer vorhandenen Basistechnologie auf, die geeignete Schnittstellen bereitstellt, um auf deren Dienstgütefunktionalität zuzugreifen. Die Middleware gliedert sich in zwei Hauptkomponenten. Eine Komponente namens *QosReservation-LinkMgr* tätigt die Reservierung auf dem Medium entsprechend der aktiven Dienstgüteklasse. Bei einer Veränderung der Dienstgüteklasse findet eine Neuverhandlung statt. Für jede Dienstgüteverbindung wird eine eigene Prozessinstanz innerhalb der Komponente angelegt, welche selbstständig die vorhandenen Resourcen auf einer Verbindung an die einzelnen Applikation gemäß ihrer Priorität aufteilt.
Die Komponente *QosMonitoringTransfer* überwacht sämtliche Verbindungen des Kommunikationssystems. Falls auf einer dieser Verbindungen Dienstgüte reserviert worden ist, erfolgt bei jeder Änderung der Ressourcen ein Feedback an die darüberliegende Komponente. Desweiteren

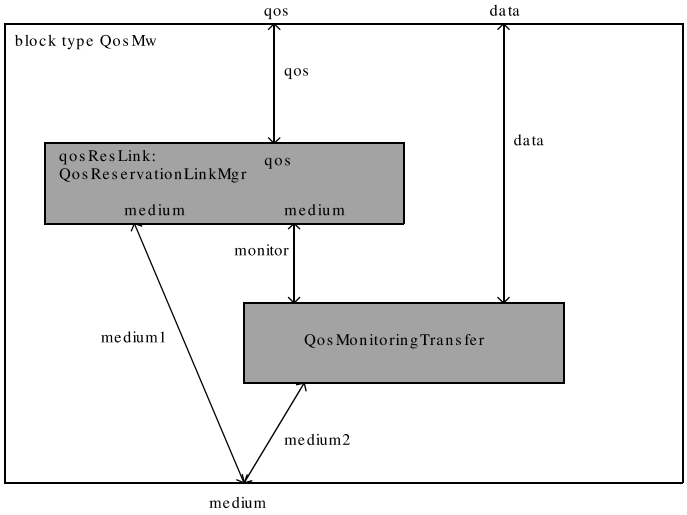

Abbildung 5. Struktur der applikationsunabhängigen Middleware

leitet sie die adressierten Pakete an den Basisdienst weiter. Je nach verwendeter Basistechnologie müssen die Pakete geeignet gescheduled bzw. gemultiplext werden. Ferner nimmt *QosMonitoringTransfer* auch Pakete vom Basisdienst entgegen und verteilt diese an die entsprechenden Applikationen.
Die einzelnen Protokollfunktionalitäten wurden auch in der applikationsunabhängigen Middleware als Mikroprotokolle entwickelt und integriert.

3.5 Basistechnologie

Um die Verfügbarkeit und Zuverlässigkeit des Mediums detailliert abschätzen zu können und Reservierungen in der Basistechnologie zu unterstützen, muss QoS innerhalb des Kommunikationssystem vertikal über Schichtengrenzen hinweg integriert werden. Erst wenn auch die Basistechnologie über QoS-Support verfügt, kann diese eine Reservierung des Mediums berücksichtigen. Grundsätzlich bieten sich auf dieser Schicht zwei Ansätze an: die Verwendung von Prioritäten zur Differenzierung unterschiedlicher Dienstgüteklassen und/oder die Reservierung von Ressourcen. Da sich beide auf der Ebene der Basistechnologie nur auf jeweils einen einzelnen Hop beziehen, ist die notwendige Integration zwischen Basistechnologie und der Routing- und Transportkomponente der Middleware offensichtlich.
Die innerhalb des Demonstrators verwendete Basistechnologie baut auf WLAN-kompatibler Hardware (IEEE 802.11b [7], Prism Chipsatz (Conexant Systems)) auf. Der 802.11b-Standard sieht mit der *Point Coordination Function (PCF)* bereits einen Mechanismus zur Unterstützung von Dienstgüte vor. Da diese PCF jedoch nicht mit multihop-adhoc-Routing harmoniert und bislang nicht in handelsüblichen WLAN-Firmwares

implementiert ist, verzichten wir auf deren Einsatz. Der Standard IEEE 802.11e [8] sieht neben einer Erweiterung der PCF auch die Einführung von Prioritäten vor. Jede Priorität lässt sich mit einem Satz von *Inter-Frame Spacings* assoziieren und verändert dadurch das Medienzugriffsverhalten basierend auf der Priorität eines Rahmens. Die Anwendung dieser Prioritäten ließe sich besser in unser Dienstgüteschema integrieren, jedoch scheitert auch dies momentan an der fehlenden Impementierung der Dienstgüteunterstützung.
Wir verwenden daher – aufbauend auf MACA/PR [9] – die verteilte Unterteilung des Mediums in Slots [10]. In jedem freien Slot kann ein Sender einen RTS-CTS Dialog initiieren, um Kollisionen zu vermeiden und gleichzeitig eine Reservierung von Slots für zukünftige Übertragungen vorzunehmen (*piggyback reservation*). Da unser MACA/PR* einerseits auf WLAN aufbaut (und nicht nahtlos in die WLAN-Firmware integriert wird) und andererseits Multicast unterstützen soll, waren mehrere Probleme zu lösen:

- Das Carrier-Sense und Acknowledgement Problem: Verwendet man das in WLAN vorgesehene Carrier Sensing, so ergeben sich Exposed Node Probleme. Das Carrier Sensing verbietet, dass zwei Sendeknoten (von RTS/Daten bzw. CTS/Ack) in Reichweite liegen, obwohl eine entsprechende und sinnvolle Reservierung in MACA/PR möglich war. MACA/PR* verbietet folglich Reservierungen, bei denen ein Sender oder Empfänger in Reichweite des aktuellen Senders/Empfängers liegt.
- Erweiterung für Multicast: Um eine Multicast-Kommunikation zu ermöglichen, wurde eine unvollständige Slotreservierung sowie ein Acknowledgement für den Multicast-Fall eingeführt. In beiden Fällen kann ein Multicastempfänger ausgewählt werden, der für das CTS bzw. Ack verantwortlich ist.

Das resultierende MACA/PR*-Protokoll wurde zunächst mit Hilfe von MSCs und SDL spezifiziert und mit Telelogic Tau simuliert. Die Implementierung erfolgte schließlich in Form eines Linux-Kernelmoduls per Hand nach einem zuvor festgelegten Schema, basierend auf einem modifizierten linux-wlan-ng Treiber [11]. In der aktuellen Fassung werden lediglich 10 Slots pro Sekunde untersützt, durch die Einbeziehung von Echtzeiterweiterungen [12,13] des Linux-Kerns ist dies jedoch noch auf über 100 Slots/Sekunde steigerungsfähig.

4 Zusammenfassung und Ausblick

In diesem Beitrag wurde ein maßgeschneidertes Kommunikationssystem mit Dienstgüteunterstützung vorgestellt. Es wurden wichtige Dienstgütemechanismen identifiziert und deren Anwendung bei der Fernsteuerung eines Luftschiffes erläutert. Die einzelnen Protokollfunktionalitäten wurden in SDL als Mikroprotokolle spezifiziert und die Middleware nach anwendungsabhängigen und anwendungsunabhängigen Teilen strukturiert. Dadurch konnten schon während der Entwicklung des Systems einige Teile wiederverwendet und in die Mikroprotokollbibliothek aufgenommen

werden. Außerdem wird es durch die Verwendung eines komponentenbasierten Ansatzes in Zukunft möglich sein, dieses Kommunikationssystem um eine Vielzahl von Funktionalitäten zu erweitern. Unter anderem sind bereits die Zusammenführung der vorgestellten Architektur und die eines Multicast-Routingverfahrens in Arbeit. In diesem Zusammenhang ist ein neues Adressierungsschema geplant, das eine in Adhoc-Netzen effiziente Unicast/Multicast-Adressierung unterstützt. Als weiteres Ziel ist die vollständige Integration von Dienstgüte in ein Multi-Hop-Routingprotokoll geplant.

Literatur

1. I. Fliege, A. Geraldy, R. Gotzhein, P. Schaible: *A Flexible Micro Protocol Framework.* In Proccedings of 4th SDL and MSC Workshop SAM'04, Ottawa, Canada, June 2004.
2. R. Gotzhein, F. Khendek: *Conception avec Micro-Protocoles.* Colloque Francophone sur l'Ingenierie des Protocoles, Montreal, Canada, 2002
3. C. Aurrecoechea, A.T. Campbell, L. Hauw: *A Survey of QoS Architectures*, ACM/Springer Verlag Multimedia Systems Journal, Special Issue on QoS Architecture, Vol. 6, No. 3, Mai 1998
4. C. Webel: *Entwicklung und Integration von QoS-Mikroprotokollen zur Steuerung eines Fluggerätes über WLAN.* Diplomarbeit, Technische Universität Kaiserslautern, 2004
5. H. Kopetz: *Real-Time Systems – Design Principles for Distributed Embedded Applications.* Kluwer Academic Publishersa, 1997
6. ITU-T Recommendation Z.100 (11/99): *Specification and Description Language (SDL).* International Telecommunication Union (ITU), 1999
7. IEEE Std. 802.11b: *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification: Higher-Speed Physical Layer Extensions in the 2.4 GHz Band*, 1999
8. IEEE Draft 802.11e: *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specification: Medium Access Control (MAC) Enhancements for Quality of Service (QoS)*, 2001
9. C.R. Lin, M. Gerla: *Asynchronous Multimedia Multihop Wireless Networks*, IEEE INFOCOM '97, 1997
10. A. Geraldy, R. Gotzhein, D. Schmidt: *Improvement, Extension, Specification and Implementation of MACA/PR*, Technischer Bericht Nr. 5 (2004), Berichte des Forschungsschwerpunkts Ambient Intelligence, Universität Kaiserslautern, 2004
11. AbsoluteValue Systems: *The linux-wlan Project*, http://www.linux-wlan.org
12. FSMLabs Inc.: *RTLinuxFree*, http://www.fsmlabs.com/products/openrtlinux/
13. Dipartimento di Ingegneria Aerospaziale - Politecnico di Milano: *RTAI – Realtime Applicatin Interface*, http://www.aero.polimi.it/~rtai/