Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Seminarausarbeitung

Kai-Uwe von Deylen

Ein Service oriented Architecture basiertes Konzept zur Kommunikation im Automobil

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Kai-Uwe von Deylen

Ein Service oriented Architecture basiertes Konzept zur Kommunikation im Automobil

Seminarausarbeitung eingereicht im Rahmen der Veranstaltung MS

im Studiengang Master Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Eingereicht am: 5. Januar 2015

Inhaltsverzeichnis

1 Einleitung, Motivation und Fragestellung

Zur Kommunikation zwischen Steuergeräten (electronic control unit, ECU) in Automobilen kommt derzeit eine Vielzahl von Bussystemen zum Einsatz, welche speziell auf die Bedingungen im Fahrzeug abgestimmt sind (vgl. [1, 2, 3]). Die verschiedenen Anwendungsdomänen besitzen jeweils für sie typische Anforderungen und Eigenschaften bezüglich der ausgetauschten Daten, sodass die ECUs einer Domäne jeweils über eine entsprechende Technologie verbunden sind. Bspw. ist in der Domäne „Chassis“ das Versenden von kleinen Datenmengen mit harter Echtzeit- bei minimaler Latenzanforderung typisch, wohingegen im Bereich „Infotainment“ größere Datenmengen auftreten, deren Zeitanforderungen jedoch weniger streng sind (vgl. [4]). Das stetige Erweitern und Anpassen an sich ändernde Kommunikationscharakteristika dieser über die Zeit gewachsenen Vernetzung führt jedoch zu folgenden Problemen (vgl. [1, 5]).

Domänenübergreifende Funktionen erfodern die Kommunikation über die Grenzen einzelner Bussysteme hinaus. Da sich die Bussysteme z. T. stark in ihren Anforderungen und der Art der Datenübertragung unterscheiden, sind hierzu komplexe Gateways notwendig und die Komplexität des Gesamtsystems steigt signifikant. Zudem verursacht jede Änderung der Kommunikation, z. B. durch Integration einer neuen Funktion, ein Anpassen der Gateways und eine Rekonfiguration der Vernetzung, sodass ein erheblicher Aufwand entsteht.

Neben der Komplexität stellen auch steigende Bandbreitenanforderungen, bedingt durch videoverarbeitende Fahrerassistenzsysteme, Multimediaanwendungen o. Ä., eine Schwierigkeit für das herkömmliche Fahrzeugnetzwerk dar. Um die hohen Datenraten bei gleichzeitig geringer Latenz zu ermöglichen, werden momentan Punkt-zu-Punkt-Verbindungen benötigt. Diese zusätzliche Verkabelung bewirkt eine direkte Steigerung der Produktionskosten, sodass die „Kosten pro Bandbreite“ bei der Vernetzung eine bedeutende Rolle spielen.

[1, 6, 7] schreiben, dass die Verwendung von Ethernet mit Echtzeiterweiterungen zur Lösung der beschriebenen Probleme beitragen kann. Diesbezüglich wurden bisher hauptsächlich die Echtzeiteigenschaften der Erweiterungen untersucht (vgl. [8, 9, 10, 11]). Momentan wird Ethernet bereits für ausgewählte Anwendungen wie Diagnose oder als Punkt-zu-Punkt-Verbindungen, z.B. zu Kameras, eingesetzt. Weitere Schritte bei der Einführung können zunächst die Verwendung als Backbone des gesamten Fahrzeugnetzes und anschließend das Ersetzen der bisherigen Bussysteme hin zu einem homogenen Ethernet Netzwerk sein (vgl. [1, 2]).

Die aktuelle Struktur der im Fahrzeug übertragenen Daten entspricht dem Paketaufbau des jeweiligen Bussystems, ohne dass dabei eine höhere Schicht zur Abstraktion genutzt wird. Ohne die Einführung von höheren Kommunikationsschichten und fortgeschrittenen Kommunikationsmustern bedeutet der Wechsel zu Ethernet daher lediglich eine kostengünstige Erhöhung der Bandbreite und eine Vereinfachung der Netzwerktopologie. Außerdem wird das gesamte System statisch konfiguriert

und getestet, sodass das Hinzufügen oder Ändern einer Komponente derzeit eine Gesamtintegration[1] zur Folge hat. So wird in [12] und [13] deutlich, dass eine größere Flexibilität bezüglich der Softwarekomponenten im Fahrzeug sinnvoll ist, um den enormen Aufwand der Gesamtintegration zu reduzieren.

Aus diesen Aspekten ergibt sich das Themengebiet der vorliegenden Arbeit: Die Strukturierung der Ethernet-basierten Kommunikation im Automobil. Ein SOA (Service orientierte Architektur) basiertes Konzept wird vorgestellt, welches die Integration neuer Anwendungen ohne Beeinträchtigung der Funktionalität und Integrität des bestehenden Systems erlauben soll, damit keine erneute Gesamtintegration sowie Überprüfung des kompletten Systems erforderlich ist. Die Fragestellung lautet dementsprechend: Ermöglicht das nachfolgend vorgestellte Konzept einer Service oriented Architecture auf Basis von Ethernet mit Echtzeiterweiterungen die angestrebte Flexibilität bei gleichzeitiger Einhaltung der domänenspezifischen Anforderungen?

Zunächst werden in Abschnitt 2 die Rahmenbedingungen – Eigenschaften der kommunizierten Daten, typische Kommunikationsmuster, Anforderungen an die Kommunikation und an das Gesamtsystem – als Bewertungsgrundlage für die sich anschließenden Untersuchungen aufgeführt. In Abschnitt 3 werden verschiedene Lösungsansätze aufgezeigt, woraufhin der in dieser Arbeit verfolgte Ansatz erläutert. Das Vorgehen bei der Analyse und Bewertung ist in Abschnitt 4 beschrieben. Abschließend wird in Abschnitt 5 eine Zusammenfassung der relevanten Aspekte der Ausarbeitung und ein Ausblick gegeben.

2 Anforderungen und Bewertungsgrundlage

Je nach Fahrzeugklasse besteht das verteilte System Auto aus über 70 ECUs (vgl. [1]), welche untereinander verschiedene Arten von Daten austauschen. [14] gruppiert die kommunizierten Daten innerhalb des Fahrzeugs in vier Kategorien und nennt jeweils typische temporale Anforderungen:

Steuerdaten: Steuerdaten sind Messwerte von Sensoren, Befehle an Aktoren oder diverse Größen in Regelkreisen. Hier treten i. d. R. kleine Datenmengen mit Paketgrößen von wenigen Bytes auf, welche jedoch die höchsten Anforderungen bzgl. Latenz besitzen. Nach [14] liegen die strengsten Anforderungen bei max. $2,5ms$ Latenz, wobei zur Unterstützung zukünftiger Anwendungen jedoch auch geringere Latenzen erfüllt werden müssen.

Echtzeit Multimedia: Diese Kategorie umfasst alle Audio- und Videostreams, deren Verarbeitung in Echtzeit erforderlich ist, wie es bspw. bei kamerabasierten Fahrerassistenzsystemen oder im Infotainment der Fall ist. Hier nennt [14] maximale Latenzen von ca. $33ms$ und eine Paketverlustrate $\leq 0{,}1\%$. Im Gegensatz zu den Steuerdaten liegt die erforderliche Bandbreite jedoch in

[1] Als Gesamtintegration wird das Zusammenführen aller Komponenten, deren Partitionierung auf die Hardware-Ressourcen und das Erzeugen der Systemkonfiguration bezeichnet.

der Größenordnung von mehreren MBit/s.

Multimedia Daten: Diese Daten dienen der Unterhaltung der Fahrgäste, wobei es sich ebenfalls um Audio sowie Videodaten handelt. In dieser Kategorie bestehen allerdings keine harten Anforderungen bezüglich maximaler Latenz sondern bezüglich Jitter. Das Ziel ist das gleichmäßige Abspielen der Unterhaltungsmedien ohne stockende Wiedergabe, sodass die Wiedergabe durchaus mit einer geringen Verzögerung starten darf, jedoch nach Wiedergabebeginn die Daten rechtzeitig zur Verfügung stehen müssen. [14] gibt in dieser Kategorie eine maximale Latenz von $100ms$ und als max. Paketverlustrate ebenfalls $\leq 0{,}1\%$ an.

Sonstige Best Effort Daten: Best Effort Daten zeichnen sich durch das nicht Vorhandensein von Anforderungen bezüglich Latenz, Jitter oder Paketverlust aus. Dies ist z.B. bei Diagnosefunktionen der Fall.

In [2] und [6] werden ebenfalls Charakteristiken der im Automobil kommunizierten Daten genannt, welche die genannten Werte im Allgemeinen bestätigen.

Wie einleitend beschrieben wurde, sind neben diesen Anforderungen auch Konzepte zur Strukturierung der kommunizierten Daten von großer Bedeutung. [15] beschreibt, dass die Anforderungen an die Systemarchitektur während der Entwicklungsphasen stark differieren. So ist zu Beginn der Entwicklung die Flexibilität und Anpassbarkeit des Systems wichtig, wohingegen bei der Evaluierung im Fahrzeug die funktionale Sicherheit an erster Stelle steht. In [13] wird darüber hinaus erläutert, dass eine funktionale Erweiterung oder Modifikation einzelner Komponenten durch die hohe Komplexität und der monolithischen Struktur des Gesamtsystems zu großen Wartungs- und Validierungsaufwand führen. An dieser Stelle bieten (1) die Erhöhung der Wiederverwendbarkeit von Komponenten, (2) eine bessere Integrationsunterstützung (vergleichbar mit Plug&Play) sowie (3) die damit verbundene Verbesserung der Erweiterbarkeit und Technologiezyklusentkopplung das größte Optimierungspotenzial (vgl. [16]). Diese drei Punkte lassen sich konkreter formulieren als Forderung nach abstrakten Schnittstellen sowie der Möglichkeit dynamisch (zur Systemlaufzeit) Komponenten hinzufügen, ausschließen oder tauschen zu können, ohne hierdurch die restliche Systemkonfiguration zu beeinträchtigen.

Die Gesamtheit dieser Anforderungen dient als Grundlage zur Überprüfung des nachfolgend erläuterten Konzeptes. Insbesondere wird untersucht, ob das Konzept die geforderte Flexibilität ermöglicht, ohne dabei die Erfüllung der Latenzanforderungen zu behindern. So darf bspw. das Hinzufügen neuer Services oder Clients nicht das bestehende System stören.

3 Ein SOA-basierter Ansatz zur Kommunikation im Automobil

In der Forschung werden verschiedene Ansätze untersucht, um die in Abschnitt 2 genannten Eigenschaften in zukünftigen Systemen im Automobil besser zu unterstützen. So wird u. a. durch

Standardisierungsinitiativen (z.B. AUTOSAR [17]) versucht, die Komplexität durch definierte Entwicklungsprozesse und Automatisierung von Entwicklungsschritten entgegenzuwirken. Das resultierende System hat jedoch eine monolithische Struktur mit fester Bindung zwischen den verwendeten Komponenten, sodass eine spätere Anpassung der Komponenten oder das Hinzufügen neuer Funktionalitäten nicht trivial ist. Daher wird in [13] ein Konzept vorgestellt, welches diese Anpassung von Komponenten während der Entwicklungszyklen ohne Gesamtintegration ermöglicht. Über die Entwicklungsphase hinaus, z. B. zur Systemlaufzeit, erlaubt es jedoch keine weitere Anpassung der Komponenten. [16] beschreibt einen Ansatz, der diese Problematik mittels einer Service orientierten Architektur adressiert, und geht dabei insbesondere auf die dynamische Orchestrierung[2] von Services ein. In diesem Kapitel wird der in dieser Arbeit verfolgte, ebenfalls Service orientierte Ansatz erläutert.

Bei Service orientierten Architekturen wird die extern nutzbare Funktionalität der Teilnehmer auf Services abgebildet. So könnte bspw. ein Autoradio einen Service zum Steuern der Wiedergabe (z.B. Lautstärke, Senderwahl) anbieten, welcher von mobilen Geräten genutzt wird. Jeder Service bietet also eine definierte Funktionalität, welche über Operationen genutzt werden kann. Eine Funktionalität kann auch das Anbieten von Informationen, z. B. Sensorwerte, sein. Um solche Informationen zu propagieren, existiert ein Eventing Mechanismus. Zur Nutzung dieser Funktionen benötigt ein Client Metainformationen, die beschreiben, wie mit dem jeweiligen Service Provider zu kommunizieren ist und welche Services mit welchen Funktionen dieser anbietet.

In [18] wird ein *Real-Time Service-Oriented Architecture* Framework im Szenario des algorithmischen Handelns[3] vorgestellt. Darin werden die verschiedenen Aspekte erläutert, welche für eine echtzeitfähige SOA relevant sind:

- Betriebssystem / Laufzeitumgebung des Service Provider sowie des Service Client
- Kommunikationsinfrastruktur
- Komposition von mehreren Services
- Umsetzung in der Middleware

Die vorliegende Arbeit stellt ebenfalls ein Echtzeit-SOA Konzept vor, befindet sich jedoch im Kontext der Kommunikation in Fahrzeugnetzen und fokussiert sich dabei auf eine echtzeitfähige Kommunikationsinfrastruktur der Services.

Um Echtzeit-Garantien für Service Operationen und Events zu ermöglichen, müssen in diesem Zusammenhang zwei Bedingungen erfüllt werden: Zum einen muss die durch die Operation verursachte Latenz begrenzt sein, d.h. die Verarbeitungs- und Kommunikationszeit muss geringer sein als die Zeitspanne, welche sich aus den temporalen Anforderungen ergibt. Zum anderen muss sichergestellt sein, dass die benötigten Ressourcen zur richtigen Zeit zur Verfügung stehen, woraus sich die Notwendigkeit von preemptiver Ressourcenzuteilung oder der Reservierung von Ressourcen ergibt. Um

[2] Orchestrierung bezeichnet das Kombinieren von Services. Dies kann bspw. die Verkettung von Services sein.
[3] Algorithmisches Handeln bezeichnet den automatischen, durch Programme ausgeführten Handel mit Wertpapieren.

dem System dynamisch neue Service Provider und Clients hinzufügen zu können, muss die Prüfung dieser Bedingungen zur Laufzeit möglich sein. In Abschnitt 3.1 werden die relevanten Komponenten beschrieben, anschließend in Abschnitt 3.2 die Kommunikationsinfrastruktur erläutert, welche diese Bedingungen erfüllen soll, und abschließend in Abschnitt 3.3 der Mechanismus zur Integration neuer Services oder Clients erläutert.

3.1 Relevante Komponenten

Die für Echtzeit-Garantien relevanten Komponenten sind in Abbildung 1 dargestellt und werden nachfolgend erläutert. Im Falle von Service Events werden die Komponenten entlang der dargestellten Reihenfolge (A-B-C-D) durchlaufen. Im Falle von Service Operationen sind dieselben Komponenten relevant, werden jedoch zunächst für die Anfrage des Clients in umgekehrter Reihenfolge und anschließend in der dargestellten Reihenfolge (Antwort des Service Providers) durchlaufen.

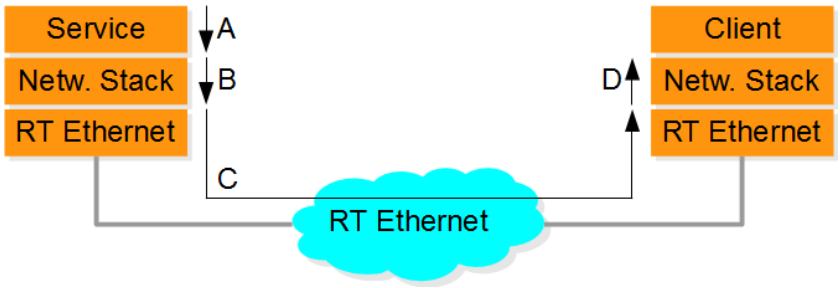


Abbildung 1: Systemübersicht mit den für Latenzgarantien relevanten Abschnitten

A Provider App
Wenn der angebotene Service durch ein Ereignis angestoßen wird, welches das Senden eines Events zur Folge hat, so muss zunächst dieses Ereignis ausgewertet und ggf. die weitere Verarbeitung abgeschlossen sein, bevor das Event versendet wird. Darüber hinaus kann auch das Detektieren des Ereignisses bereits verzögert werden, z.B. durch das Scheduling des Service Providers. Hieraus resultiert die Latenz *A*, deren Obergrenze in statischen Konfigurationen bestimmt werden kann. Für die Behandlung von Operation Requests von Clients verhält es sich analog.

B Provider Stack
Die Verarbeitung im Netzwerk Stack des Service Providers verursacht eine weitere Verzögerung *B* beim Empfang von Operation Requests oder dem Senden von Operation Replys bzw. Events. Diese Verzögerung kann per Analyse des Stacks offline ermittelt werden (vgl. [19]). Um Garantien für maximale Latenzen für das Gesamtsystem zu ermöglichen, ist es notwendig, dass Stack / Middleware sowie das darunterliegende Betriebssystem (falls vorhanden) harte

Echtzeit unterstützten. D.h. es müssen entsprechende Scheduling Strategien eingesetzt werden und Ressourcen Reservierungen möglich sein. Dies kann insbesondere im Fall von dynamisch (zur Laufzeit) hinzugefügter Komponenten anspruchsvoll sein, da somit adaptive Änderungen am Deadline-basierten Scheduling nötig sind.

C Network
Während der Entwicklung einer Software Komponente (Service oder Client), welche ähnlich einer App auf einem Smartphone zur Laufzeit installiert werden kann, sind u.Ust. keine genauen Informationen über die Netzwerk Charakteristiken (Latenz, verfügbare Bandbreite) vorhanden. Daher müssen diese Informationen zur Laufzeit von der Software Komponente ermittelt und ausgewertet werden.

D Client Stack
Die Verzögerung durch den Client Netzwerk Stack kann analog zur Verzögerung im Provider Stack bestimmt werden.

3.2 Kommunikationsinfrastruktur

Unter Kommunikationsinfrastruktur wird an dieser Stelle die Funktionsweise bzw. das Zusammenspiel der verschiedenen Kommunikationsschichten verstanden. Abbildung 2 gibt eine Übersicht über diese. Als Kommunikationsbasis dient ein Ethernet Netzwerk, welches um ein TDMA Konzept, der Priorisie-

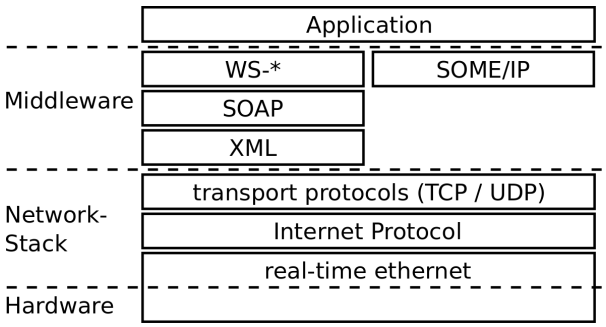


Abbildung 2: Protokollstack einer echtzeitfähigen Service orientierte Architektur

rung von Paketen sowie dem Audio Video Bridging (AVB) Standard [20] erweitert wird, um maximale Latenzen und Paketverlustraten entsprechend der Anforderungen (vgl. Abschnitt 2) zu garantieren. Daraus ergeben sich verschiedene Übertragungsklassen, welche mittels Critical Traffic ID (TDMA betreffend, kurz: CT-ID) bzw. Stream ID (AVB betreffend) priorisiert werden. Neben der Garantie bezüglich maximaler Latenz und mindestens verfügbarer Bandbreite ist auch die maximal verfügbare Bandbreite pro CT-ID bzw. Stream Klasse festgelegt, um das Blockieren niederpriorer Kommunikation zu verhindern. Die Auswirkungen der Koexistenz von TDMA-basierten Nachrichten und AVB auf die

Kommunikations-Eigenschaften wurde in [21] untersucht.

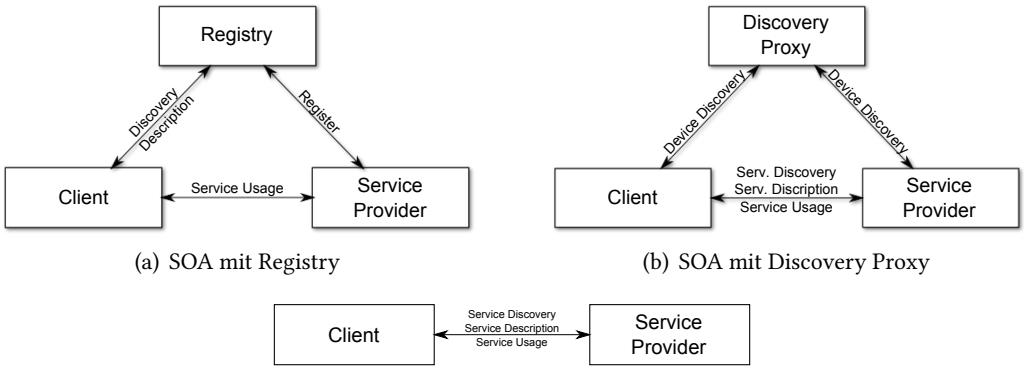
Zur Abstraktion von der Network-Access-Schicht wird das *Internet Protocol* (IP) [22] verwendet. An dieser Stelle ist eine Art Mapping nötig, um die IP Adressierung auf die Übertragung mittels Ethernet Erweiterungen abzubilden. Diese Abbildung muss zur Laufzeit anwendbar sein, um auch während des Betriebes neue Services anbieten bzw. nutzen zu können. Auf der Transportschicht kommen sowohl das verbindungslose UDP als auch das verbindungsorienterte TCP zum Einsatz, um die SOA Nutzdaten zu übertragen. Neben der SOA Kommunikation können auch andere Daten anfallen, welche ebenfalls über IP oder mittels anderer Protokolle kommuniziert werden können. Da verschiedene Services und auch verschiedene Operationen eines Services sich in ihren Kommunikations-Anforderungen unterscheiden können, wird eine Zuordnung von einzelnen Datenpaketen mittels verschiedener Kriterien zur entsprechenden CT-ID bzw. zum entsprechenden AVB Stream vorgenommen.

Hierauf basierend koexistieren Subkomponenten des Devices Profile for Web Services (DPWS, vgl. [23]) und der zu AUTOSAR gehörigen SOA, SOME/IP (vgl. [24]). SOME/IP wird aufgrund der binären Serialisierung für operative Zwecke genutzt, bspw. das Ausführen von Service Operationen oder Versenden von Events, um so eine schnelle Verarbeitung auch auf Geräten mit eingeschränkten Ressourcen zu erlauben. Da SOME/IP voraussetzt, dass die Metainformationen über Services (verfügbare Operationen, abbonierbare Events, QoS Eigenschaften) offline bekannt sind, wird für Service Discovery und Ressourcen Reservierungen hingegen der XML und SOAP basierte DPWS Teil eingesetzt. Somit ist neben einer effizienten Service Nutzung durch binäre Serialisierung auch das Hinzufügen neuer Teilnehmer oder Software Komponenten möglich, ohne dass diese Vorwissen über das Netzwerk und das bestehende System benötigen.

3.3 Discovery von Services und Ressourcenreservierung

Typischerweise erfolgt die Service Discovery in Service orientierten Architekturen über eine zentrale, allgemein bekannte Service Registry, bei welcher die Service Provider die Metadaten ihrer Services hinterlegen und Clients diese abfragen können (vgl. [25]). Es bestehen i. d. R. drei Phasen: Die *Service Discovery* bezeichnet das Suchen bzw. Abfragen von verfügbaren Services; die *Service Description* bezeichnet den Austausch von Metainformationen, welche zur Servicenutzung benötigt werden; zuletzt folgt die eigentliche Service Nutzung. In [23] wird eine für ressourcenbeschränkte Geräte angepasste Service Discovery beschrieben, welche anstelle einer zentralen Service Registry ein Verzeichnis von Service Providern (Discovery Proxy) nutzt (vgl. Abbildung 3), um die umfangreiche Registry zu umgehen. Nachdem die Service Provider sich bei dem Discovery Proxy registriert haben, können Clients dort die nötigen Informationen zu den Service Providern abrufen, um anschließend die Service-Metadaten direkt vom Service Provider zu beziehen. In dem Konzept der vorliegenden Arbeit wird ein Discovery Mechanismus ohne zentrale Komponente verwendet. Hierzu wird die Eigenschaft ausgenutzt, dass das Netzwerk im Automobil lokal und somit in seiner Größe beschränkt ist: Ein Client sendet eine

Anfrage mit Metainformationen über einen gesuchten Service sowie die Mindestanforderungen an alle Teilnehmer, von denen die Service Provider, welche einen passenden Service mit den entsprechenden Eigenschaften anbieten, auf diese Anfrage antworten (vgl. Abbildung 3).



(a) SOA mit Registry

(b) SOA mit Discovery Proxy

(c) SOA ohne zentrale Komponente

Abbildung 3: Grobe Struktur Service orientierter Architekturen

Wie in diesem Kapitel einführend beschrieben wurde, sind zwei Bedingungen vor der Service Nutzung (Empfangen von Events, Ausführen von Operationen) sicherzustellen: Die maximale Latenz muss den Anforderungen entsprechen und die benötigten Ressourcen – in diesem Fall die benötigte Bandbreite bzw. das Übertragungsmedium – müssen verfügbar sein. Um diese Voraussetzungen zur Laufzeit zu überprüfen werden folgende Informationen benötigt:

Benötigte Bandbreite Die benötigte Bandbreite ergibt sich aus der Größe der Daten und der Häufigkeit des Versendens. Im Fall von Events wird die maximale Größe sowie das minimale Intervall auf Seiten des Service Providers bestimmt und betrifft ausschließlich die Übertragung vom Service Provider zum Client. Ein Operationsaufruf besteht hingegen aus Request des Clients und Response des Service Providers, wobei das Aufrufintervall durch den Client bestimmt wird. Um zu häufiges Aufrufen durch den Client zu verhindern wird vorausgesetzt, dass die minimale Zeitspanne zwischen zwei Requests und die maximale Größe eines Requests sowie einer Response durch den Provider bestimmt werden kann (vgl. Abschnitt 3.1).

Latenzanforderung Das in dieser Arbeit beschriebene Konzept geht von der Annahme aus, dass die Latenzanforderung stets vom Client ausgeht. Wird ein Service von mehreren Clients mit unterschiedlichen temporalen Anforderungen verwendet, kann somit für jeden Client separat überprüft werden, ob die Anforderungen erfüllt werden. Im umgekehrten Fall (Latenzanforderung geht vom Service aus) müssten auch Clients mit geringeren Anforderungen, bspw. Anwendungen zur Diagnose, die möglicherweise harten Echtzeitanforderungen des Services erfüllen.

Verfügbare Bandbreite Die verfügbare Bandbreite für Pakete einer bestimmten CT-ID, also für den gesamten TDMA basierten Datenverkehr, ist sowohl Provider- als auch Clientseitig offline bekannt, sodass diese nicht zur Laufzeit kommuniziert werden muss. Bei der Registrierung von AVB Streams wird hingegen zur Laufzeit von jedem beteiligten Knoten geprüft, ob die angeforderte Bandbreite noch verfügbar ist. In beiden Fällen muss sichergestellt sein, dass die reservierte Bandbreite je Provider-Client Tupel nicht kleiner ist als die maximal benötigte Bandbreite.

max. Latenz Die Informationen zur maximalen Latenz bestehen analog zur verfügbaren Bandbreite. Für TDMA Pakete existiert dieses Wissen Provider- wie Clientseitig vor der Laufzeit, wohingegen die maximale Latenz eines AVB Streams während der Stream Reservierung bestimmt wird.

Nach der zuvor beschriebenen Service Discovery werden diese Informationen mittels WS-Agreement [26] kommuniziert, überprüft und ggf. die Ressourcen reserviert.

4 Analyse des Konzeptes und Risiken

Die Untersuchung des beschriebenen Konzeptes mit Hinblick auf die in Abschnitt 2 genannten Anforderungen erfolgt zunächst für das Verhalten im Normalfall, um die grundlegende Funktionsweise zu überprüfen. Um mit geringem Aufwand den Einfluss verschiedener Szenarien, Netztopologien und Parametrierungen betrachten zu können wird das Konzept mittels Simulation analysiert. Da das dynamische Hinzufügen neuer Services oder Clients zur Laufzeit unterstützt werden soll, sind insbesondere deren Auswirkungen auf das zuvor bestehende Systemverhalten zu untersuchen. In [27] wird das Vorgehen bei simulationsbasierten Analysen in sechs Schritte aufgeteilt, welche bei der Untersuchung des Konzeptes durchlaufen werden:

1. Problembeschreibung und Definition der Modellgrenzen/-abstraktion
2. Auswahl der Metriken und Parameter für das Modell
3. Entwurf des Modells
4. Implementierung des Modells und Verifikation der Implementation
5. Validierung des Modells bezüglich der Parameter und Metriken
6. Durchführung der Experimente am Modell und Analyse der Ergebnisse

Das Modell wird in der Simulationsumgebung OMNeT++ [28] implementiert und es werden hierbei die bereits bestehenden Frameworks für Ethernet sowie dessen Erweiterung um AVB und TDMA (INET [29] und CoRE4INET [30]) verwendet.

Jeder der genannten Punkte birgt Potenzial für systematische Fehler bei der Analyse. Das größte Risiko besteht in der Validierung des Modells, da derzeit weder ein Referenzsystem noch Referenzdaten

für eine solche Validierung zur Verfügung stehen. Darüber hinaus besteht das Risiko, dass einige Annahmen, auf denen das Konzept basiert, sich als falsch erweisen. Hier ist beispielsweise ein unerwartetes Verhalten der Netzwerk- oder Transportschicht aufgrund der Ethernet Echtzeiterweiterungen zu nennen, welches ggf. separat zu untersuchen ist. Ein mögliches Ergebnis der Simulation ist auch, dass dynamisch hinzugefügte Services/Clients zwar keine Auswirkung auf das bestehende System haben, die Verarbeitung im Netzwerkstack jedoch zu Aufwändig ist und somit die z. T. strengen Zeitanforderungen nicht eingehalten werden können. Außerdem besteht bei einer Simulation stets die Gefahr, dass wichtige Szenarien nicht abgedeckt werden und somit die Bewertung des Konzeptes falsch positiv ausfällt. Denkbar wäre hier bspw. folgendes Szenario: Das dynamische Hinzufügen eines Services während der Systemlaufzeit zeigt keine Beeinträchtigung des Systems. Nach einem Systemneustart könnten sich jedoch einige Voraussetzungen durch das Hinzufügen des Services geändert haben, sodass eine Beeinträchtigung erst nach einem Systemneustart eintritt. Die Erkennung dieses Verhaltens mittels Simulation wäre insbesondere dann schwierig, wenn die Beeinträchtigung von Race Conditions abhängt.

5 Zusammenfassung und Ausblick

In der vorliegenden Arbeit wurde ein Konzept zur Strukturierung der ethernetbasierten Kommunikation im Automobil vorgestellt, welches Echtzeiterweiterungen für Ethernet im Rahmen einer Service orientierten Architektur vorsieht. Dazu wurden zunächst die Motivation sowie der Bedarf für ein solches Konzept aufgezeigt, welche hauptsächlich aus der steigenden Komplexität der Netzwerke im Fahrzeug und dem steigenden Bedarf an Flexibilität entstehen. Es wurden Anforderungen an das aus dem Konzept entstehende System identifiziert, welche als Bewertungsgrundlage der simulationsbasierten Untersuchung dienen.

Einige Punkte sind jedoch offen geblieben, sodass sie z. T. Risiken darstellen und zukünftig nachgeholt oder zumindest beachtet werden müssen: Die Implementierung eines entsprechenden Prototypen würde die Validierung des Modells ermöglichen und könnte gleichzeitig als eine Referenzimplementierung dienen. Desweiteren ist eine worst-case Betrachtung sinnvoll, wenn die Simulationsergebnisse als valide angesehen werden können und das Konzept bestätigen. Dies könnte z. B. mit einer formalen Analyse des Konzeptes erfolgen, welche wiederum eine Modellbildung und somit ein eigenes Fehlerpotenzial einschließt. Nicht zuletzt ist auch ein Vergleich mit anderen Konzepten, welche die genannte Problematik adressieren, von Bedeutung.

Literatur

[1] Lucia Lo Bello. The case for ethernet in automotive communications. *ACM SIGBED Review*, 8(4): 7–15, December 2011. ISSN 15513688. doi: 10.1145/2095256.2095257. URL http://dl.acm.org/citation.cfm?id=2095256.2095257.

[2] Hyung-Taek Lim, Lars Völker, and Daniel Herrscher. Challenges in a future IP/ethernet-based in-car network for real-time applications. In *Proceedings of the 48th Design Automation Conference*, page 7, New York, New York, USA, June 2011. ACM Press. ISBN 9781450306362. doi: 10.1145/2024724.2024727. URL http://dl.acm.org/citation.cfm?id=2024724.2024727.

[3] T Streichert and M Traub. *Elektrik/Elektronik-Architekturen im Kraftfahrzeug*. 2012. ISBN 9783642254772. URL http://link.springer.com/content/pdf/10.1007/978-3-642-25478-9.pdf.

[4] Mehrnoush Rahmani, Joachim Hillebrand, Wolfgang Hintermaier, Richard Bogenberger, and Eckehard Steinbach. A Novel Network Architecture for In-Vehicle Audio and Video Communication. In *2007 2nd IEEE/IFIP International Workshop on Broadband Convergence Networks*, pages 1–12. IEEE, May 2007. ISBN 1-4244-1297-8. doi: 10.1109/BCN.2007.372741. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4238838.

[5] Alexander Camek, Christian Buckl, Pedro Sebastiao Correia, and Alois Knoll. An Automotive Side-View System Based on Ethernet and IP. In *2012 26th International Conference on Advanced Information Networking and Applications Workshops*, pages 238–243. IEEE, March 2012. ISBN 978-1-4673-0867-0. doi: 10.1109/WAINA.2012.66.

[6] Lucia Lo Bello. Novel trends in Automotive Networks: A perspective on Ethernet and the IEEE Audio Video Bridging. In *IEEE 19th International Conference on Emerging Technologies & Factory Automation (ETFA)*, 2014.

[7] Till Steinbach, Franz Korf, and Thomas C. Schmidt. Real-time Ethernet for automotive applications: A solution for future in-car networks. In *2011 IEEE International Conference on Consumer Electronics -Berlin (ICCE-Berlin)*, pages 216–220. IEEE, September 2011. ISBN 978-1-4577-0233-4. doi: 10.1109/ICCE-Berlin.2011.6031843.

[8] Giuliana Alderisi, Alfio Caltabiano, Giancarlo Vasta, Giancarlo Iannizzotto, Till Steinbach, and Lucia Lo Bello. Simulative assessments of IEEE 802.1 Ethernet AVB and Time-Triggered Ethernet for Advanced Driver Assistance Systems and in-car infotainment. In *2012 IEEE Vehicular Networking Conference (VNC)*, pages 187–194. IEEE, November 2012. ISBN 978-1-4673-4996-3. doi: 10.1109/VNC.2012.6407430. URL http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=6407430.

[9] Giuliana Alderisi, Giancarlo Iannizzotto, and Lucia Lo Bello. Towards IEEE 802.1 Ethernet AVB for Advanced Driver Assistance Systems: A preliminary assessment. In *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation (ETFA 2012)*, pages 1–4. IEEE, September 2012. ISBN 978-1-4673-4737-2. doi: 10.1109/ETFA.

2012.6489775. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6489775.

[10] Rodney Cummings, Kai Richter, Rolf Ernst, Jonas Diemer, and Arkadeb Ghosal. Exploring Use of Ethernet for In-Vehicle Control Applications: AFDX, TTEthernet, EtherCAT, and AVB. Technical Report 1, April 2012. URL http://saepcelec.saejournals.org/content/5/1/72.abstract.

[11] Till Steinbach, Hyung-Taek Lim, Franz Korf, Thomas C. Schmidt, Daniel Herrscher, and Adam Wolisz. Tomorrow's In-Car Interconnect? A Competitive Evaluation of IEEE 802.1 AVB and Time-Triggered Ethernet (AS6802). In *2012 IEEE Vehicular Technology Conference (VTC Fall)*, pages 1–5. IEEE, September 2012. ISBN 978-1-4673-1881-5. doi: 10.1109/VTCFall.2012.6398932. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6398932.

[12] Edward C Nelson, K Venkatesh Prasad, and Ingolf H Krüger. Service-Based Software Development for Automotive Applications. 2004.

[13] Alexander Zeeb. Plug and Play Solution for Autosar Software Components. *ATZelektronik worldwide*, 2012. URL http://link.springer.com/article/10.1365/s38314-012-0069-2.

[14] Rainer Steffen, Richard Bogenberger, Joachim Hillebrand, Wolfgang Hintermaier, Andreas Winckler, and Mehrnoush Rahmani. Design and Realization of an IP-based In-car Network Architecture. In *Proceedings of the First Annual International Symposium on Vehicular Computing Systems*, Gent, BELGIUM, 2008. ICST. ISBN 9789639799271. doi: 10.4108/ICST.ISVCS2008.3543. URL http://eudl.eu/?id=3543.

[15] Jan Gacnik, Oliver Häger, and Marco Hannibal. A Service-oriented system architecture for the human centered design of intelligent transportation systems. In *Proceedings of European Conference on Human Centered Design for Intelligent Transport Systems*, pages 175—-181. HUMANIST NoE, 2008. URL http://www.conference.noehumanist.org/document/Proceedings-HUMANIST.pdf.

[16] Jörg Küfen, Janek Hudecek, and Lutz Echstein. Automotive Service Oriented System Architecture – Ein neues Architekturkonzept und sein Potential für zukünftige Fahrzeugsysteme. In *Automotive meets Electronics 2014 (GMM-FB 78)*. VDE Verlag, 2014. URL https://www.vde-verlag.de/proceedings-en/453580019.html.

[17] AUTOSAR. AUTOSAR AUTomotive Open System ARchitecture, 2013. URL http://www.autosar.org/.

[18] Mark Panahi, Weiran Nie, and Kwei-Jay Lin. A Framework for Real-Time Service-Oriented Architecture. *2009 IEEE Conference on Commerce and Enterprise Computing*, pages 460–467, July 2009. doi: 10.1109/CEC.2009.78. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5210760.

[19] Ivan Cibrario Bertolotti and Tingting Hu. Real-time performance of an open-source protocol stack for low-cost, embedded systems. In *ETFA2011*, pages 1–8. IEEE, September 2011. ISBN 978-1-4577-0017-0. doi: 10.1109/ETFA.2011.6059000. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6059000.

[20] IEEE Standard for Local and metropolitan area networks–Audio Video Bridging (AVB) Systems. *IEEE Std 802.1BA-2011*, 2011.

[21] Philipp Meyer, Till Steinbach, Franz Korf, and Thomas C. Schmidt. Extending IEEE 802.1 AVB with time-triggered scheduling: A simulation study of the coexistence of synchronous and asynchronous traffic. In *2013 IEEE Vehicular Networking Conference*, pages 47–54. IEEE, December 2013. ISBN 978-1-4799-2687-9. doi: 10.1109/VNC.2013.6737589. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6737589.

[22] RFC 791: Internet Protocol, 1981. URL http://www.ietf.org/rfc/rfc791.txt.

[23] Elmar Zeeb, Andreas Bobek, Hendrik Bohn, and Frank Golatowski. Service-Oriented Architectures for Embedded Systems Using Devices Profile for Web Services. *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07)*, pages 956–963, 2007. doi: 10.1109/AINAW.2007.330. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4221181.

[24] Lars Völker. SOME/IP – Die Middleware für Ethernet-basierte Kommunikation. *HANSER automotive networks*, pages 17–19, 2013.

[25] Mark Endrei, Jenny Ang, Ali Arsanjani, Sook Chua, Philippe Comte, Pal Krogdahl, Min Luo, and Tony Newling. *Patterns: Service Oriented Architecture and Web Services*. 2004. ISBN 073845317X. URL http://www.redbooks.ibm.com/redbooks/pdfs/sg246303.pdf.

[26] Alain Andrieux, Karl Czajkowski, Asit Dan, Kate Keahey, Heiko Ludwig, Toshiyuki Nakata, Jim Pruyne, John Rofrano, Steve Tuecke, and Ming Xu. Web Services Agreement Specification (WS-Agreement). *Open Grid Forum*, 2007. URL http://www.ogf.org/documents/GFD.107.pdf.

[27] Klaus Wehrle, Mesut Günes, and James Gross. *Modeling and tools for network simulation*. Springer, 2010. ISBN 978-3-642-12330-6.

[28] OMNeT++. URL http://www.omnetpp.org.

[29] INET Framework. URL http://inet.omnetpp.org/.

[30] Till Steinbach, Hermand Dieumo Kenfack, Franz Korf, and Thomas C. Schmidt. An extension of the OMNeT++ INET framework for simulating real-time ethernet with high accuracy. pages 375–382, March 2011. URL http://dl.acm.org/citation.cfm?id=2151054.2151120.

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 5. Januar 2015 Kai-Uwe von Deylen