Norbert Balbierer

Energiemanagement Ethernet-basierter Fahrzeugnetze

Energiemanagement Ethernet-basierter Fahrzeugnetze

Norbert Balbierer

Universitätsverlag Ilmenau
2018

Impressum

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Angaben sind im Internet über http://dnb.d-nb.de abrufbar.

Diese Arbeit hat der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität Ilmenau als Dissertation vorgelegen.

Tag der Einreichung:	10. Oktober 2016
1. Gutachter:	Univ.-Prof. Dr. rer. nat. Jochen Seitz (Technische Universität Ilmenau)
2. Gutachter:	Prof. Dr. rer. nat. Thomas Waas (Ostbayerische Technische Hochschule Regensburg)
3. Gutachter:	Prof. Dr. techn. Markus Kucera (Ostbayerische Technische Hochschule Regensburg)
Tag der Verteidigung:	10. November 2017

Technische Universität Ilmenau/Universitätsbibliothek
Universitätsverlag Ilmenau
Postfach 10 05 65
98684 Ilmenau
http://www.tu-ilmenau.de/universitaetsverlag

readbox unipress
in der readbox publishing GmbH
Am Hawerkamp 31
48155 Münster
http://unipress.readbox.net/

ISBN 978-3-86360-175-1 (Druckausgabe)
URN urn:nbn:de:gbv:ilm1-2017000691

Titelfoto: photocase.com | AlexFlint

Kurzfassung

Die Einführung von Ethernet als Vernetzungstechnologie für neue Fahrzeuggenerationen eröffnet ein weites Feld an Möglichkeiten für vernetzte Funktionen und Systeme im Automobil. Die wachsende Anzahl elektronischer Systeme heutiger Fahrzeuge hat jedoch auch einen zunehmend deutlichen Anteil am gesamten Energiebedarf und damit einhergehend an Kraftstoffverbrauch und CO_2-Emission. Gleichzeitig steigt das Bestreben, Fahrzeuge effizienter zu gestalten, um Energieverbrauch und Emissionen zu reduzieren. Neben der Erhöhung der Energieeffizienz der einzelnen Komponenten ist das selektive Abschalten vernetzter Steuergeräte während des Betriebs des Fahrzeugs (Teilnetzbetrieb) ein Mittel zur Optimierung der Leistungsaufnahme der Bordelektronik. In der vorliegenden Arbeit werden Konzepte zur Umsetzung von Teilnetzbetrieb für Ethernet-basierte Netzwerke vorgeschlagen. Um Steuergeräte durch das Netzwerk aufwecken zu können wurde ein ruhestromloser Weckempfänger entwickelt, realisiert und getestet. Zur Koordination des Teilnetzbetriebs wurden zwei Netzwerkmanagement-Protokolle erarbeitet, implementiert und evaluiert. Die beschriebenen Konzepte wurden in einem Testfahrzeug unter realen Bedingungen eingesetzt und erprobt.

Abstract

The introduction of Ethernet as a next-generation networking technology for passenger and commercial vehicles enables a multitude of networked functions and systems. However, the growing number of on-board electronics also leads to a significant rise in overall power consumption and thus fuel usage and CO_2 emissions. At the same time, it is crucial that cars become more efficient, in order to reduce overall energy consumption and emission of greenhouse gases. Besides improving energy efficiency of electronic components in general, selective activation and deactivation of networked devices during operation of the car (partial networking) is a newly introduced way of optimizing power consumption. This work presents a set of concepts that enable partial networking for Ethernet networks. In order to wake up devices over the network, a zero-power wake-up receiver has been developed, built as a prototype and tested. Two network management protocols that allow coordination of partial network states have been developed and implemented. All discussed approaches have been evaluated on board of a testing vehicle within a real in-car environment.

Danksagung

Zum Gelingen einer Dissertation tragen sehr viel mehr Menschen bei, als nur der Autor. Sie helfen mit ihrer Erfahrung, mit ihrem Rat und, am wichtigsten, mit ihrer Unterstützung durch alle Höhen und Tiefen eines so langwierigen Vorhabens. Ihnen möchte ich an dieser Stelle meinen Dank widmen.

Ich danke Prof. Dr. Jochen Seitz, meinem Doktorvater. Er hat mich die gesamte Zeit über mit gutem Rat und hilfreichen Anregungen betreut und begleitet. Besonders möchte ich auch Prof. Dr. Thomas Waas meinen Dank aussprechen, der als mein Zweitgutachter und Betreuer an der OTH Regensburg immer ein offenes Ohr für mich hatte. Unsere Gespräche waren stets ein Quell der Inspiration. Ebenso danke ich Prof. Dr. Markus Kucera, meinem Drittgutachter an der OTH Regensburg, für sein gutes Feedback, seine Hinweise sowie gemeinsame Konferenzbesuche.

Bei Continental in Regensburg durfte ich vielen Menschen begegnen, die durch ihre Unterstützung zu dieser Arbeit beigetragen haben. Ich danke besonders Josef Nöbauer, damals Gruppenleiter im Bereich Vernetzung, der mir diese Arbeit erst ermöglicht hat. Insbesondere möchte ich auch Dr. Helge Zinner danken. Unsere Diskussionen und ein gemeinsames Beschreiten des Wegs der Promotion haben vieles einfacher und angenehmer gemacht. Ebenso gilt mein Dank den Studenten, die durch ihre Abschlussarbeiten und Praktika zum Gelingen dieser Dissertation beigetragen haben und allen anderen beteiligten Personen, die hier nicht namentlich augeführt sind.

An dieser Stelle möchte ich mich bei Dr. Marcus Baum, Daniel Lammering, Max Seidenstücker und Dennis Degueldre für ihre Unterstützung auch abseits der Arbeitsthemen bedanken.

Mein spezieller Dank gilt Dipl.-Ing. Robert Karl. Deine Korrekturen waren eine große Hilfe für mich.

Ganz besonders danken möchte ich meinen Eltern, Gerhard und Heidrun Balbierer, dafür, dass sie mich in allen Lebenslagen mit unendlicher Geduld unterstützen. Ihr habt mir alles ermöglicht und seid stets für mich da.

Die letzte Danksagung gilt Ulrike Karl, meiner Lebensgefährtin. Danke, dass Du mich darin unterstützt hast, diese Arbeit fertigzustellen und nicht aufzugeben, auch wenn der Berg unüberwindbar schien. Danke, dass Du da bist.

Norbert Balbierer, Regensburg den 5. Februar 2018

Inhaltsverzeichnis

1 Effizienzsteigerung in der Automobilindustrie

1.1 Europas Klimapolitik

1.1.1 Vereinbarte Ziele

Europa hat sich ehrgeizige Ziele bei dem wichtigen Thema Klimaschutz gesteckt und sich dazu verpflichtet, bis zum Jahr 2050 die Treibhausgasemissionen um 80 bis 95 Prozent gegenüber dem Basisjahr 1990 zu reduzieren. In der Mitteilung KOM(2011) 112 der Europäischen Kommission wurde 2011 ein „Fahrplan für den Übergang zu einer wettbewerbsfähigen CO_2-armen Wirtschaft bis 2050“ vorgestellt, um Strategien für die einzelnen Wirtschaftssektoren zu nennen, die zum Erreichen des Gesamtziels beitragen können [28]. Die Staffelung sieht vor, dass Treibhausgasemissionen bis zum Jahr 2020 zunächst um 20 Prozent verringert werden. 2030 soll die Senkung bereits 40 Prozent, im Jahr 2050 schließlich 80 Prozent betragen, was dem mit der internationalen Staatengemeinschaft vereinbarten Ziel Europas entspricht.

1.1.2 Rolle des Europäischen Straßenverkehrs

Aus dem vierten Sachstandsbericht des IPCC (Intergovernmental Panel on Climate Change) geht hervor, dass 2004 der Transportsektor 26 Prozent des globalen Energieverbrauchs verursacht, und sich für 23 Prozent (ca. 6 Gt CO_2-Äquivalent) der weltweiten, aus energetischer Nutzung (Verkehr, Heizung, Industrie und Stromerzeugung) stammenden Treibhausgasemissionen (ca. 23 Gt CO_2-Äquivalent) verantwortlich zeichnet [60]. Dies entspricht einem Anstieg der durch den Transportsektor bedingten CO_2-Emissionen um 27 Prozent gegenüber 1990. Drei Viertel des transportbedingten Energieverbrauchs (77 EJ) sind bedingt durch den weltweiten Straßenverkehr mit

Personenkraftwagen und leichten Nutzfahrzeugen als maßgeblichem Anteil (44 Prozent) [60].

In Europa betrug 2006 der Anteil des Verkehrssektors an den energetisch genutzten CO_2-Emissionen 26 Prozent und damit mehr als der durch die Sektoren Haushalt (19 Prozent) und Industrie (16 Prozent) verursachte. Mit 39 Prozent liegt der Anteil des Sektors Energieerzeugung darüber [81]. Der Gesamtbeitrag Europas zu den weltweiten CO_2-Emissionen betrug 2013 ca. $4,2$ Gt CO_2-Äquivalent bei einer Weltproduktion von etwas unter 35 Gt CO_2-Äquivalent [65]. Europa zählt damit zu den sechs größten Kohlendioxidemittenten weltweit, zusammen mit China, den Vereinigten Staaten, Indien, Russland und Japan [65]. Die Europäische Kommission schätzt, ausgehend von ihrer Analyse, das Potential zur Verringerung von Treibhausgasen in anderen Wirtschaftssektoren als dem Verkehrssektor (z. B. Stromerzeugung und Industriesektor) zwar höher ein, dennoch muss auch der transportbedingte Treibhausausstoß bis 2050 um 60 Prozent gegenüber 1990 reduziert werden, um das gesteckte Gesamtziel erreichen zu können [29]. Als Zwischenziel für 2030 wird eine Senkung um 20 Prozent gegenüber dem Jahr 2008 angestrebt [29], was immer noch einer Erhöhung der Treibhausausstöße um 8 Prozent gegenüber 1990 entspricht. Das bedeutet, dass in nächster Zeit die verkehrsbedingten CO_2-Emissionen nicht reduziert werden können, sondern lediglich deren Anstieg, bedingt durch massiven Zuwachs von Personen- und Güterverkehr während des letzten Jahrzehnts, durch entsprechende Maßnahmen abgeflacht werden kann.

Um das Ziel im Verkehrssektor erreichen zu können, setzt die Europäische Kommission in ihrer Analyse auf Innovation in drei Hauptfaktoren:

- Verwendung von umweltschonender Energie durch neue Kraftstoffe und Antriebskonzepte
- bessere Nutzung von Netzen und sicherer Betrieb durch Informations- und Kommunikationssysteme.
- Fahrzeugeffizienz durch neue Kraftstoffe, Antriebskonzepte und neues Design

Im *Weißbuch Verkehr 2011* [29] werden hierfür unter anderem mögliche Maßnahmen aufgezeigt und zusammengefasst. Das Dokument beschränkt sich dabei nicht nur auf Treibhausgasemissionen, sondern versucht generell Europas Vision eines nachhaltigen und wettbewerbsfähigen Verkehrssystems darzustellen. Wichtige Rollen spielen dabei die Vereinheitlichung des europäischen Verkehrsraumes, um durch Unterschiede in den einzelnen Mitgliedsstaaten bedingte Engpässe oder Unzulänglichkeiten zu beheben,

sowie ein effizienterer Mobilitätsgedanke im Personen- und Güterverkehr. Neue Mobilitätskonzepte stützen sich auf eine bessere Verzahnung der einzelnen Mobilitätsnetze, d.h. ein reibungsloses Ineinandergreifen von Fern- und Nahverkehr, um Überlastungen und Staus zu reduzieren. Daneben wird der technischen Innovation eine wichtige Rolle zugeschrieben. Von der Verwendung umweltschonender Energie durch neue Antriebskonzepte und Kraftstoffe und von der Erhöhung der Fahrzeugeffizienz verspricht man sich, dem Erreichen der Emissionsziele im Verkehrssektor einen Schritt näher zu kommen.

1.1.3 Auswirkungen auf die Automobilindustrie

Diese Zielsetzung spiegelt sich auch in den Emissionsrichtlinien der EU (Europäische Union) wieder. Mit Verordnung Nr. 443/2009 [30] setzte die Union den europäischen Automobilherstellern Ziele für die Emission ihrer Neuwagenflotten. Die Überschreitung des jeweiligen Grenzwertes ist mit hohen Strafabgaben verbunden. Dies soll einen Anreiz schaffen, in innovative Technologien zu investieren, und damit Flottenverbrauch und -emission zu reduzieren. So gilt ab 2012 das Ziel von 120 g/km CO_2 für die durchschnittliche Emission aller europäischen Neuzulassungen. Bereits 2020 liegt dieser Wert bei 95 g/km CO_2. Die Zielvorgabe für die spezifische Emission hängt dabei von der Fahrzeugmasse ab. Bei schweren Fahrzeugen wird mehr Potential zur Verringerung von Emissionen gesehen als bei leichteren, weswegen für diese auch eine prozentual höhere CO2-Minderung festgelegt wird. Bezogen auf die mittleren Emissionen der EU-Flotte 2006 müssen demnach Hersteller bei Fahrzeugen mit einem Gewicht von 1,6 Tonnen deren CO_2-Emissionen um 24 % (bzw. 45 g/km) reduzieren, während die Minderung bei Fahrzeugen mit einem Gewicht von einer Tonne lediglich 12 % (bzw. 16 g/km) beträgt. Für einen Hersteller ergibt sich damit die Zielvorgabe aus der Mittelung der spezifischen Emissionsziele über die gesamte Flotte, bzw. aus dem Durchschnittsgewicht der Flottenfahrzeuge. Überschreiten die spezifischen Emissionen diese Zielvorgabe, so fallen signifikante Strafzahlungen pro neu zugelassenem Fahrzeug an. Ab dem Jahr 2019 berechnen sich diese aus der Formel (Überschreitung $\times$ 95 Euro pro g/km) $\times$ Anzahl neuer Fahrzeuge. Bis 2019 sind die Kompensationszahlungen gestaffelt nach der Höhe der Überschreitung, mit einem maximalen Satz von 95 Euro ab drei Gramm über dem Zielwert. Bei einem Hersteller mit einem Absatz von zwei Millionen Fahrzeugen pro Jahr bedeutet bereits eine Überschreitung von einem Gramm ab 2019 Kompensationszahlungen in Höhe von 190 Millionen Euro. Der Verband der Automobilindustrie (VDA) sieht diese als unverhältnismäßig und überzogen an, da

die Höhe der Zahlungen ein Vielfaches größer ist als in anderen Industriezweigen bei Überschreitungen. Dennoch muss die Automobilindustrie nun Technologien und Wege finden, um Flottenverbrauch und -emission auf den Zielwert zu bringen. Dies erstreckt sich auf viele Bereiche der Fahrzeugtechnik.

1.2 Ansätze zur Reduktion von Kraftstoffverbrauch und Emissionen

Getrieben von Anreizen und Verordnungen der EU ist der Trend zur Effizienzsteigerung und Verbrauchsreduktion in der Automobilindustrie klar zu erkennen. Dem gegenüber steht aber auch der Wunsch nach agilen Motoren, Komfort und Sonderausstattungen wie Fahrerassistenz und Konnektivität.

1.2.1 Optimierung an vielen Stellen

Motoren werden sparsamer, bieten aber dennoch hohe Fahrleistungen. Dieses als *Downsizing* bekannte Konzept beruht darauf, größere Motoren durch kleinere Motoren mit Turbolader zu ersetzen und durch intelligentes Motormanagement in der Motorsteuerung deren Effizienz weiter zu erhöhen. Dadurch sinkt der Verbrauch zum einen durch die Gewichtsreduktion des Motors, die sich in der Gesamtfahrzeugmasse niederschlägt, und zum anderen durch die Verkleinerung des Hubraumes, wodurch weniger Kraftstoff eingespritzt wird. Durch den Einsatz von Turboladern, die Verwendung besserer Werkstoffe und die Motorsteuerung können mit diesen Motoren bei geringerem Verbrauch gleiche, teilweise sogar bessere Fahrleistungen erzielt werden. Die Verbesserung des Wirkungsgrades des Generators ist ein weiterer Schritt zur Steigerung der Effizienz. Während in der Literatur weitgehend von Wirkungsgraden um die 60 % [17] gesprochen wird, bieten Hersteller heute neue Generationen von Generatoren an, die auf geringeren Verbrauch und niedrigere CO_2-Emission hin optimiert sind, und Wirkungsgrade bis zu 76 % erreichen. Die Reduktion des Gesamtgewichtes stellt ebenfalls einen wichtigen Hebel dar. Automobilhersteller verwenden immer mehr leichte Werkstoffe im Fahrzeugdesign, wie etwa Magnesium. Auch beim Kabelbaum des Fahrzeugs wird versucht, Gewicht zu reduzieren. Heutige Oberklassefahrzeuge wie z. B. die S-Klasse von Mercedes besitzen einen Kabelbaum von drei Kilometer Länge und 3800 Steckverbindungen [17]. Durch Vernetzungskonzepte und moderne Netzwerktechnologien kann die Anzahl

benötigter Leitungen im Fahrzeug reduziert werden, und damit Gesamtmasse eingespart werden. Die Einführung von Energierückgewinnungssystemen (Rekuperation) trägt ebenfalls zur Senkung des Verbrauchs bei. Dabei wird die Ladestrategie der Batterie so verändert, dass diese insbesondere dann geladen wird, wenn das Fahrzeug kinetische Energie abbaut, also beim Bremsen oder im Schubbetrieb. Im Gegensatz zum Laden bei normaler Fahrt kostet dies keinen Treibstoff. Dies ermöglicht derzeit eine Senkung des Kraftstoffverbrauchs um 1 %-4 % [17]. Aus dem Rennsport bekannte KERS-Systeme (Kinetic Energy Recovery System), die je nach Ausführung zusätzliche Generatoren und Kondensatoren oder mechanische Schwungräder verwenden, werden heute noch nicht bei Straßenfahrzeugen verwendet. Vielleicht ändert sich dies jedoch in der Zukunft. Eine immer wichtigere Rolle spielen alternative Antriebskonzepte. Die Elektrifizierung des Antriebsstranges trägt dazu bei, den Flottenverbrauch und damit die Flottenemissionen zu reduzieren. Angefangen bei Hybridfahrzeugen bis hin zu rein elektrisch angetriebenen Autos fließen deren Stückzahlen direkt in die Berechnung der Flottenemission ein und reduzieren diese somit. Dies ist zum einen eine Möglichkeit, die bei den Verbrennerfahrzeugen notwendigen Einsparungen etwas zu entschärfen, zum anderen wird damit aber auch implizit die Forschung an innovativen neuen Antriebskonzepten gefördert, was ebenfalls erklärtes Ziel der EU-Kommission ist. Zu den aufgezählten Maßnahmen kommen ergänzende Konzepte wie beispielsweise die Start-Stopp-Automatik hinzu. Diese schaltet bei Stillstand den Motor ab, etwa an roten Ampeln oder im Stau. Die adaptive Zylinderabschaltung deaktiviert bei normaler Fahrt einzelne Zylinder, und schaltet diese bei Leistungsanforderung adaptiv hinzu, auch dies reduziert den Durchschnittsverbrauch. Nicht zuletzt werden auch Systeme entwickelt, die den Fahrer zu sparsamer Fahrweise zu motivieren sollen. So gibt es beispielsweise Schaltpunktanzeigen, die den optimalen Schaltzeitpunkt signalisieren, so dass der Fahrer die Gänge effizient nutzt. Ein weiteres Beispiel sind Fahrwahlschalter, die verschiedene Fahrzeugkonfigurationen ermöglichen, um so gleichzeitig ein ökonomisches Fahrverhalten zu bieten, bei Bedarf aber auch ein sportliches und dynamisches. Hierbei werden etwa Gaspedalkennlinie, Automatikgetriebe und Fahrwerk jeweils unterschiedlich abgestimmt.

Gleichzeitig ist aber auch der Trend zu immer mehr elektronischen Funktionen zu erkennen. Fahrer schätzen elektronische Fahrassistenzsysteme, umfassende Sensorik und damit verbundene automatisierte Funktionen, ein breites Spektrum an Multimedia-Systemen und Internet-Konnektivität. Dies gipfelt in einem der Megatrends der Automobilwelt, dem automatisierten Fahren. Ziel dabei ist es, über eine Vielzahl an Sensorik (Radar, LIDAR, Kamerasysteme, ...) ein umfassendes Abbild der Umgebung

und der Situation zu bekommen, und mittels elektronisch ansteuerbarer Aktuatorik (Lenkung, Bremse, Motorsteuerung) automatisiert im Straßenverkehr zu fahren und somit den Fahrer von dieser Aufgabe zu entbinden. Dies führt zu einer Zunahme an Bordelektronik, die in den nächsten Jahren nicht weniger werden wird. Damit wird die Bordelektronik, also die Gesamtheit der verbauten Steuergeräte, Sensoren und Aktuatoren, zu einem immer größeren Verbraucher in der Energiebilanz. Die oben dargestellten Konzepte fokussieren sich weitestgehend auf Motor und Antriebsstrang, die elektrisch/elektronische Architektur bietet jedoch auch immer mehr Potential zur Effizienzsteigerung.

1.2.2 Der Wert eines Watts

Um den Einfluss der elektrischen/elektronischen Architektur im Fahrzeug auf dessen Energiebilanz zu verstehen, ist es zunächst notwendig, den *Wert* elektrischer Energie zu betrachten und damit der Frage nachzugehen, was ein Watt elektrische Leistung im Fahrzeug an Mehrverbrauch und Emissionen verursacht.

Vom Kraftstoff zur elektrischen Energie

Elektrische Verbraucher werden bei Fahrzeugen mit Verbrennungsmotor aus einem Drehstromgenerator gespeist. Dieser wird über die Welle des Motors angetrieben und wandelt einen Teil der mechanischen Energie in elektrische Energie um. Der Motor wiederum gewinnt diese Energie aus der Verbrennung des Treibstoffes. Jeder Schritt in dieser Kette ist verlustbehaftet. Dies reduziert den Wirkungsgrad, bzw. erhöht die Menge an Treibstoff, die zur Gewinnung eines Watts an elektrischer Energie verbrannt werden muss. Aktuelle Fahrzeugmotoren erreichen Wirkungsgrade von 36 % (Ottomotoren) bzw. 43 % (Dieselmotoren) [17]. Dabei handelt es sich um Bestwerte, die im

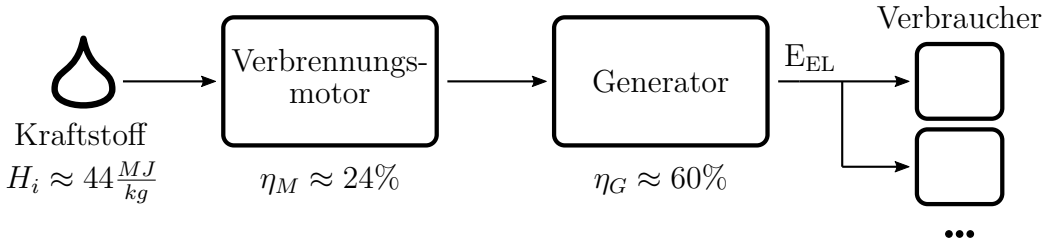


Abbildung 1.1: Verlustbehaftete Wirkungskette bei der Erzeugung elektrischer Energie

mittleren Drehzahlbereich und unterhalb der Vollastkurve erzielt werden können [17]. Der Rest der im Kraftstoff enthaltenen Energie geht in thermischen und mechanischen Verlusten auf. Klauenpolgeneratoren, die im 12 V-Bordnetzbereich zur Erzeugung von Drehstrom eingesetzt werden, erzielen Wirkungsgrade von ca. 60 % [17]. Damit ergibt sich ein gesamter Wirkungsgrad von ca. 22 % (Ottomotor) bzw. 26 % (Dieselmotor) bei der Umwandlung von Kraftstoff zu elektrischer Energie im Bordnetz. Die in einem Liter Treibstoff gespeicherte Energie kann aus dem Heizwert und der Dichte berechnet werden. Benzin hat einen Heizwert von 44 MJ/kg bei einer Dichte von ca. $0,72$ kg/dm^3 [77]. In einem Liter Benzin ist demnach die Energie $31,7$ MJ bzw. $8,8$ kWh enthalten. Bei Diesel sind diese Werte ähnlich: der Heizwert beträgt $43,2$ MJ/kg, die Dichte ist mit ca. $0,78$ kg/dm^3 etwas höher [77]. Ein Liter Diesel enthält damit die Energie $33,7$ MJ ($9,4$ kWh).

Die Umwandlung eines Liters Kraftstoff mit dem kombinierten Wirkungsgrad von Motor und Generator liefert also ca. $1,9$ kWh (Benzin) bzw. $2,4$ kWh (Diesel) an elektrischer Energie verbraucherseitig in das 12 V-Bordnetz. Umgerechnet bedeutet dies einen Kraftstoffverbrauch von etwa $0,5$ ml Benzin ($0,4$ ml Diesel) pro Stunde für ein Watt elektrischer Leistung im Bordnetz. Bei der Verbrennung von einem Liter Benzin entstehen $2,37$ kg Kohlenstoffdioxid, bei Diesel sind es $2,65$ kg [52]. Auf die oben berechnete, für ein Watt elektrischer Leistung notwendige Kraftstoffmenge bezogen werden also pro Stunde $1,18$ g (Benzin) bzw. $1,06$ g (Diesel) an Kohlenstoffdioxid erzeugt. Die Durchschnittsgeschwindigkeit im NEFZ (Neuer Europäischer Fahrzyklus) beträgt $33,6$ km/h [17]. Ein Watt elektrische Leistung im Fahrzeug verursacht also bezogen auf den NEFZ CO2-Emissionen von 35 mg/km und einen Mehrverbrauch von $1,5$ ml / 100 km. Wie später noch dargestellt wird, liegt die elektrische Leistungsaufnahme der Steuergeräte in heutigen Fahrzeugen bei einer Größenordnung von 400 bis 700 Watt. Könnte man beispielsweise einhundert Watt davon einsparen, so ließen sich Schadstoffemission und Verbrauch um $3,5$ g/km CO_2 beziehungsweise $0,15$ l / 100 km reduzieren.

Energiekosten

Aus dem Zusammenhang zwischen einem Watt elektrischer Leistung und Treibstoffverbrauch bzw. Emissionen lässt sich ableiten, welche Kosten elektrische Verbraucher im Auto verursachen. Für den Fahrzeughalter spielt neben dem Mehrverbrauch und den daraus resultierenden Kraftstoffkosten die Kraftfahrzeugsteuer eine Rolle. Bei ab

Juli 2009 zugelassenen Personenkraftwagen (PKW) bezieht die Steuer nicht nur den Hubraum, sondern auch die CO_2-Emissionen des Fahrzeugs mit ein (§8 KraftStG). Der Jahressteuersatz berechnet sich aus 2 Euro je angefangenen 100 cm^3 bei Ottomotoren, (9,50 Euro bei Dieselmotoren) und 2 Euro pro Gramm Kohlendioxidemission über dem festgesetzten Grenzwert. Dieser beträgt seit Januar 2014 95 g/km. Die durchschnittlichen Flottenemissionen der Automobilhersteller lagen 2006 im Bereich 140 g/km bis 180 g/km [81], das mittelfristige EU-Ziel liegt bei 130 g/km, und damit noch weit über der Besteuerungsgrenze von 95 g/km. Damit bewegt sich der emissionsbedingte Anteil der Kraftfahrzeugsteuer zwischen 90 Euro und 170 Euro. Jedes eingesparte Gramm reduziert diesen Betrag, den der Fahrzeugbesitzer zu entrichten hat. Ein Watt ist damit für den Fahrzeugbesitzer umgerechnet 7 ct wert. Eine Einsparung von 100 Watt (entsprechend einer Reduktion um $3,5$ g/km) senkt die Kraftfahrzeugsteuer demnach um sieben Euro.

Deutlich höher sind die Auswirkungen jedoch für den Automobilhersteller. Wie in Abschnitt 1.1.3 dargestellt, wirkt sich ein Gramm mit 95 Euro an Kompensationszahlungen pro Fahrzeug aus. Ein Watt besitzt demnach für den Automobilhersteller einen Wert von 3,33 Euro pro Fahrzeug. Bei einem Jahresvolumen von beispielsweise zwei Millionen Neufahrzeugen entspräche dies jährlichen Kosten von 6,7 Millionen Euro pro Watt. Eine Einsparung von hundert Watt brächte damit eine Kostensenkung von 670 Millionen Euro pro Jahr mit sich.

2 Energiebedarf im Bordnetz

2.1 Die elektrische/elektronische Architektur eines Fahrzeugs

Wie die Organismen auf der Erde sich von einem Einzeller zu einem komplexen Organismus aus Milliarden oder mehr Zellen entwickelt haben, hat sich die Elektronik im Automobil von einer kleinen Glühbirne hin zu einem komplexen Netzwerk verteilter elektronischer und elektrischer Systeme gewandelt. Die ersten (serienmäßig produzierten) Automobile um die Jahrhundertwende (z. B. Ford Model T, 1910) verfügten lediglich über elektrische Lampen. 1959 wurde die Transistorzündung eingeführt, 1967 folgte die elektronische Benzineinspritzung. In den darauffolgenden Jahren nahm die Anzahl elektrischer und elektronischer Systeme rapide zu – zunächst insbesondere im Bereich der Steuer- und Regelsysteme, wie etwa die digitale Motorelektronik (1981), das elektronische Gaspedal (1988) und die Antriebs-Schlupf-Regelung (1990). Einen Meilenstein stellt die Entwicklung der dynamischen Stabilitätskontrolle (1993) dar, die bis heute wohl viele Unfälle vermieden und die Sicherheit im Straßenverkehr deutlich gesteigert hat. Neben den Steuer- und Regelsystemen, welche für die eigentliche Fahraufgabe relevant sind, entstanden auch zahlreiche elektronische Komfort-Systeme, wie zum Beispiel die elektrische Sitzverstellung, elektrisch öffen- und schließbare Heckdeckel, Funkverriegelungen, und Keyless-Go-Systeme. Zunehmend spielt auch Kommunikation und Information eine wichtige Rolle im Automobil. Telefonie, Anbindung ans Internet, Navigation und Unterhaltungssysteme bringen eine Vielzahl neuer elektronischer Geräte ins Fahrzeug. Die Anbindung von Smartphones, Tablets und Audioplayern gewinnt mehr und mehr an Bedeutung, da viele Fahrer ihre persönlichen Medien während der Fahrt genauso nutzen möchten, wie zu Hause oder beim Sport. Einen weiteren Innovationsschub stellen Fahrassistenzsysteme dar: Abstandsregelung, Notbremsassistenten oder automatische Spurhaltung werden durch vernetzte Sensorik- und Rechnersysteme realisiert. Die Vision vom Automatisierten Fahren setzt diesen

Gedanken fort. Derzeit arbeiten viele Automobilhersteller und Zulieferer an deren Realisierung.

Elektronik spielt also eine bedeutende Rolle in der modernen Fahrzeugtechnik. Mehr als 80 % aller neuen Innovationen werden durch elektronische Komponenten beeinflusst [17]. Die allermeisten dieser Funktionen werden durch dedizierte elektronische Steuergeräte (Electronic Control Unit, ECU) realisiert. Dies sind eingebettete Rechner mit Prozessoren, anwendungsspezifischen integrierten Schaltkreisen, Speicher, Versorgungsbeschaltung, Netzwerkschnittstellen, Sensorik und Aktuatorik. Heutige Oberklassefahrzeuge besitzen teilweise bereits bis zu hundert Steuergeräte, bei einem Gesamtdurchschnitt von fünfzig Steuergeräten bei europäischen Automobilherstellern [54]. Im Folgenden soll zunächst ein Überblick über die unterschiedlichen Funktionsbereiche der Fahrzeugelektronik gegeben werden. Anschließend wird insbesondere die Vernetzung der elektronischen Komponenten im Fahrzeug, die ein Kernelement der Architektur darstellt, betrachtet. Schließlich wird auch ein kurzer Überblick über AUTOSAR (*Automotive Open Systems Architecture*) gegeben. AUTOSAR ist ein Softwarestandard der Automobilindustrie, der die für die Interaktion der verschiedenen Softwarefunktionen nötige Infrastuktur und geeignete Schnittstellen schafft.

2.1.1 Funktionale Bereiche

Die elektronischen Systeme eines Fahrzeugs und mit ihnen die Steuergeräte gliedern sich in unterschiedliche funktionale Bereiche [70]. Der Bereich Antriebsstrang umfasst Systeme wie die elektronische Motorsteuerung, die Getriebesteuerung, Turbolader, Einspritzsysteme oder antriebsbezogene Sensorik. Aktive und passive Sicherheit bezeichnet Systeme, die der Sicherheit von Insassen, anderen Verkehrsteilnehmern und des Fahrzeugs selbst dienen. Dazu gehören unter anderem Rückhaltesysteme, Airbags oder Fußgängerschutzsysteme, also Systeme, die im Falle eines Unfalles Personenschaden verhindern oder minimieren sollen (passive Sicherheit). Fahrdynamische Systeme, wie das Elektronische Stabilitätsprogramm, das Antiblockiersystem (ABS) oder der Notbremsassistent, helfen dabei Unfälle von vornherein zu verhindern (aktive Sicherheit). Klimaautomatik, Innenraumbeleuchtung oder die elektrische Sitzverstellung fallen in den Bereich Komfort. Im Bereich Karosserieelektronik sind beispielsweise Zugangssysteme, Fahrzeug-Gateways, Karosseriesteuergeräte, Beleuchtung, Türsteuergeräte oder Fensterheber angesiedelt. Multimediasysteme, Internet-Konnektivität, Navigation oder die Anbindung mobiler Multimediageräte fallen in den Bereich In-

fotainment, ein in der Branche geläufiges Kunstwort, welches sich aus *Information* und *Entertainment* zusammensetzt und den Einsatzzweck der entsprechenden Komponenten beschreibt. Die immer mehr an Bedeutung gewinnenden Fahrassistenzsysteme (engl. *Advanced Driver Assistance Systems, ADAS*) bilden einen eigenen Bereich. Zu ihnen zählen Spurhalteassistenten, Verkehrszeichenerkennung oder Abstandsregelung sowie die zugehörige Sensorik (Radar, Kameras). Viele Funktionen im Fahrzeug sind heute bereichsübergreifend [70]. So nutzen beispielsweise ADAS-Systeme Steuergeräte aus den Bereichen Antriebsstrang und Sicherheit um zu bremsen, zu beschleunigen oder zu lenken. Die Vision vom automatisierten Fahren erweitert dies um weitere Komponenten aus dem Infotainment-Bereich (Konnektivität, Navigation, Lokalisierung) und eigene Steuergeräte auf denen die Fahrfunktionen realisiert sind.

2.1.2 Vernetzung im Automobil

Der Austausch von Daten zwischen den Steuergeräten wurde immer wichtiger und ist heute nicht mehr wegzudenken und erfordert ein fahrzeugweites Kommunikationsnetzwerk. Die Vernetzung nahm ihren Anfang mit dem Wunsch, Steuersignale (z. B. Tastendrücke) auf einer Busleitung zu multiplexen, anstatt jedem Signalgeber eine eigene Leitung im Kabelbaum zu spendieren. Schnell entstanden daraus neue Anwendungsfälle, wie etwa verteilte Regelsysteme, oder die gemeinsame Nutzung von Sensordaten durch mehrere Systeme. Die nächsten Schritte waren dem Streaming von Audiodaten im Multimedia-Netzwerk und die Übertragung digitaler Daten von Kameras, Radarsensoren und anderer Datenquellen. In den unterschiedlichen Einsatzgebieten sind verschiedene, anwendungsspezifische Vernetzungstechnologien entstanden, welche heute die heterogene Netzwerklandschaft moderner Fahrzeuge prägen. Im Folgenden werden diese kurz beschrieben.

Controller Area Network

CAN (Controller Area Network) wurde von Bosch entwickelt (1986), um neue Funktionalität in Fahrzeugarchitekturen zu ermöglichen. Die damit einhergehende Reduktion an Kabelbaumlänge und -gewicht durch das nun mögliche Multiplexing von mehreren Signalen auf ein- und derselben Busleitung war dabei nur Nebeneffekt [20]. Den ersten Einsatz von CAN in einem Serienfahrzeug stellt die Vernetzung von Zündung, Einspritzung, dem elektronischen Gaspedal und der Antriebs-Schlupf-Regelung im Mercedes

W140 1992 dar. Heute besitzt nahezu jedes in Europa hergestellte Fahrzeug mindestens einen, in der Regel sogar mehrere CAN-Busse.

CAN ist ein elektrisches Bussystem mit geteiltem Medium. Alle Teilnehmer (Steuergeräte) sind mit demselben differentiellen Leitungspaar verbunden. Die Datenübertragung erfolgt NRZ-kodiert (Non-return-to-zero), eine logische Eins entspricht einem rezessiven Zustand der Busleitung, bei dem der Bustreiber des Transceivers[1] hochohmig geschaltet wird. Dabei stellt sich auf beiden Busleitungen CAN_H und CAN_L etwa die halbe Versorgungsspannung ein. Bei einer logischen Null werden die Busleitungen CAN_H und CAN_L durch den Bustreiber aktiv auf positive bzw. negative Versorgungsspannung gezogen und man spricht von einem dominanten Buszustand. Da also rezessive Bits (Eins) hochohmig sind, können sie durch dominante Bits (Null) überschrieben werden. Auf diese Weise erfolgt die Arbitrierung beim Zugriff auf das gemeinsam genutzte Medium. CAN-Botschaften besitzen eine eindeutige Identifikation (ID), die zu Beginn des Pakets gesendet wird. Nachrichten mit einer niedrigeren ID gewinnen also die Arbitrierung, sobald zwei Steuergeräte gleichzeitig zu senden versuchen, da die niedrigere ID ein dominantes Bit an jener Stelle besitzt, an der die höhere ID das erste rezessive Bit enthält. Der Knoten, der die unterlegene Nachricht senden möchte, erkennt dies, und stellt das Senden ein, um es nach Abschluss der höherprioren Botschaft erneut zu probieren. Dieses Verfahren wird CSMA/CR (Carrier Sense Multiple Access/Collision Resolution) genannt. CAN unterstützt Datenraten von bis zu 1 Mbit/s (*High-Speed*-CAN). In Bereichen, in denen keine hohe Bandbreite erforderlich sind, kommt auch *Low-Speed-CAN* mit einer Datenrate von bis zu 125 kbit/s zum Einsatz.

Local Interconnect Network

1998 wurde der LIN-Bus (Local Interconnect Network) eingeführt, der in einer Kooperation mehrerer Automobilhersteller und Zulieferer zur preisgünstigen Vernetzung mit niedrigen Datenraten entwickelt wurde. Das LIN-Konsortium setzte sich aus den Automobilherstellern BMW, Volkswagen, Audi, Mercedes-Benz und Volvo Cars sowie aus den Zulieferern Freescale und Mentor Graphics zusammen und veröffentlichte 2005 die LIN 2.0 Spezifikation [54]. 2013 stellte das Konsortium die Arbeit ein und übergab die LIN 2.2a Spezifikation an die ISO (International Standards Organization), wo sie als ISO-17987 [49] standardisiert wird [51].

[1] Sender/Empfänger, von engl. *Transmitter/Receiver*

LIN zielt primär auf die kostengünstige Anbindung intelligenter Sensoren, Bedienelemente oder Aktoren an Steuergeräte ab, um beispielsweise Befehle, Steuersignale oder Datenwerte zu übertragen, wobei keine hohen Datenraten oder komplexe Protokolle notwendig sind[54]. Auch LIN ist ein elektrisches Bussystem mit geteiltem Medium. Die Datenübertragung erfolgt nicht wie bei CAN differentiell, sondern massebezogen. Damit kommt der LIN-Bus mit nur einem Draht (den Rückkanal stellt die Fahrzeugmasse dar) aus, was auch dazu beiträgt, dass die Technologie sehr günstig ist. Elektrisch wird der Bus als *Wired OR* (engl. für verdrahtetes ODER) geschaltet. Der Transceiver jedes LIN-Knotens ist mittels eines *Open-Drain*-Ausgangs mit der gemeinsamen Busleitung verbunden, über den er die Spannung der Busleitung auf Masse ziehen kann, um ein dominantes Bit (logisch Null) zu senden. Eine logische Eins entspricht einem rezessiven Bit, bei dem der Ausgang hochohmig ist, und die Busleitung unbeeinflusst bleibt. Der Buszugriff erfolgt bei LIN über ein Master/Slave-Verfahren. Es gibt auf dem LIN-Bus einen Master und mehrere Slaves. LIN-Slaves senden niemals selbstständig Datenrahmen, sondern antworten lediglich dem LIN-Master. Dieser koordiniert damit die gesamte Bus-Kommunikation und den Zugriff der Slaves auf das Medium. LIN hat aufgrund seiner Einfachheit in vielen Anwendungsfällen seinen festen Platz in der Automobilindustrie gefunden.

FlexRay

FlexRay – ein Bussystem mit höherer Datenrate als CAN (10 Mbit/s) und deterministischen Eigenschaften – entstand im Rahmen des FlexRay-Konsortiums, welches aus den Automobilunternehmen BMW, Daimler AG, General Motors, Volkswagen und Bosch bestand, und sich nach Überführung der FlexRay-Spezifikation in einen ISO-Standard (ISO-17458-1 bis -5) wieder auflöste. Das Ziel von FlexRay war es, im Automobil deterministische, echtzeitfähige und fehlertolerante Kommunikation zu ermöglichen. Dies wurde insbesondere durch den Wunsch nach *X-by-Wire* Systemen erforderlich, bei denen die Fahrzeugsteuerung (Gas, Bremse, Lenkung) rein elektronisch erfolgt [54].

Media Oriented Systems Transport

Im Multimedia- und Informationsbereich (Infotainment) etablierte sich zunächst der von der 1998 gegründeten MOST-Cooperation entwickelte MOST-Bus (Media Oriented Systems Transport). Die Kernmitglieder der MOST-Cooperation (MOST Co) sind

Audi, BMW, Daimler, Harman und Microchip Technology [62]. MOST wird im Automobil als optisches Ringnetzwerk mit einer Datenrate von ca. 22,58 Mbit/s (MOST25) bzw. 147,5 Mbit/s (MOST150) eingesetzt, um (unter anderem) Displays, das Radio, digitale Tuner, Verstärker, Aktivlautsprecher und das Navigationssystem miteinander zu vernetzen [34]. Des Weiteren existiert eine Variante mit elektrischer Bitübertragungsschicht (MOST50) und der doppelten Datenrate von MOST25.

MOST ist ein synchrones Netzwerk, bei dem alle Teilnehmer zu einem Ring verschaltet sind. Frames werden mit einer Frequenz von 44,1 kHz oder 48 kHz auf dem Ring versendet. Ein Knoten der als *Timing Master* dient, beginnt dabei zyklisch den Frame mit dem Senden einer Präambel, durch die sich alle Knoten (*Timing Slaves*) auf den Systemtakt synchronisieren. Der Frame besitzt einen Bereich (Kanal) für Kontrolldaten, einen für synchrone Daten und einen für asynchrone Daten. Synchrone Daten stellen beispielsweise Audio-Samples dar die synchron mit der Systemfrequenz (z. B. 44,1 kHz) zwischen einer Audio-Quelle und einer Audio-Senke übertragen werden. Im synchronen Kanal werden über Zeitbereichsmultiplex (TDM, *Time Division Multiplexing*) den einzelnen Sendern feste Positionen innerhalb des Frames zum Senden ihres Samples zugeordnet. Darüber können mehrere Verbindungen quasi-gleichzeitig synchrone Datenströme übertragen. Im asynchronen Kanal werden Paketdaten übertragen, in der Regel Protokolle höherer Schichten. Der Zugriff erfolgt hier mittels *Token Ring*. Der Kontrolldatenkanal dient zur Übertragung von Netzwerk- und Knotensteuerungsdaten. Eine Kontrollnachricht ist dabei auf mehrere MOST-Frames (16 bei MOST25) verteilt, da nur 2 Byte des Frames für den Kontrollkanal vorgesehen sind. Im Vergleich zum synchronen Kanal steht damit dem Kontrollkanal nur eine vergleichsweise niedrige Datenrate zur Verfügung. Die Aufteilung zwischen synchronem und asynchronem Kanal ist zur Entwurfszeit wählbar. Die gesamte Framelänge beträgt bei MOST25 64 Byte, bei MOST50 128 Byte und bei MOST150 384 Byte.

Im Gegensatz zu den anderen Netzwerktechnologien im Automobil erstreckt sich MOST über alle sieben Schichten des OSI-Schichtenmodells (Open Systems Interconnect). Dabei wird ein funktionsorientierter Ansatz verfolgt. MOST definiert auf der Anwendungsschicht Funktionsblöcke (engl. *Function Block* oder kurz *FBlock*) als Abstraktion echter Geräte oder Dienste. Diese verfügen über Funktionen, welche durch entsprechende Kontrolldaten im Kontrolldatenkanal aufgerufen werden können. Sie können synchrone oder asynchrone Daten senden und empfangen. MOST ist damit ein gutes Beispiel für ein Bussystem, welches aus einer Anwendung heraus auf deren Anforderungen hin entwickelt und zugeschnitten wurde, im Gegensatz zu generischeren

Netzwerktechnologien, welche sich auf den Datentransport beschränken und möglichst unabhängig von der darüberliegenden Anwendung sind. Das MOST-System ist ausführlich in [34] beschrieben.

Ethernet

In jüngster Zeit hielt Ethernet Einzug in die Fahrzeugvernetzung. Der Zuwachs an Komplexität und Anzahl elektronischer Systeme im Fahrzeug geht einher mit immer weiter wachsenden Anforderungen an Datenrate, Skalierbarkeit und Kompatibilität der Vernetzungstechnologien. Bisherige Bussysteme stoßen hier an ihre Grenzen. Da CAN, MOST und FlexRay Bussysteme mit geteiltem Medium und damit geteilter Bandbreite sind, ist die Zunahme an Steuergeräten im Automobil problematisch. Beispielsweise wird FlexRay heute hauptsächlich wegen seiner gegenüber CAN zehnfach höheren Datenrate eingesetzt. Seine Echtzeit- und Redundanzeigenschaften spielen in der Realität eine untergeordnete Rolle. Ethernet bietet hohe Datenraten bei dedizierter Bandbreite und ist durch seine weite Verbreitung sowohl in der Informationstechnologie als auch in der Automatisierungstechnik oder in der Luftfahrt eine attraktive Technologie für das Bordnetzwerk der nächsten Generationen. Den endgültigen Durchbruch in der Automobilindustrie erlebte Ethernet mit der Einführung der Bitübertragungsschicht-Technologie *BroadR-Reach* von der Firma *Broadcom*, durch die 100 Mbit/s Vollduplex-Übertragung über ein einzelnes, ungeschirmtes, verdrilltes Adernpaar ermöglicht wurde – unter Einhaltung der EMV-Anforderungen der Automobilindustrie. Mit dem Standard IEEE (Institute of Electrical and Electronics Engineers) 802.3 100BASE-TX (Fast Ethernet) wäre dies nur mittels geschirmter Netzwerkkabel möglich gewesen, die aufgrund ihres Gewichts, ihrer Größe und der durch die Schirmung hohen Kosten keinen breiten Einsatz im Automobil gefunden hätten.

BroadR-Reach fand großen Anklang bei Zulieferern und Automobilherstellern. Es entstand eine branchenweite Gemeinschaft, die sich mit dem Thema beschäftigte und ihre neuesten Erkenntnisse auf jährlichen Technologietagungen (*Ethernet & IP Automotive Technology Day*) untereinander austauschte. Aus dieser Gemeinschaft heraus gründeten BMW, NXP und Broadcom im November 2011 das Industriekonsortium *OPEN Alliance* (*One Pair EtherNet*), in dem die Automobilindustrie technische Diskussionen, Entscheidungen und Anforderungen gemeinsam führt und erarbeitet [66]. Die *OPEN Alliance* hatte zum Ziel, die damals noch proprietäre *BroadR-Reach*-Technologie zusammen mit den Halbleiterherstellern in der Automobilindustrie zu etablieren und

Ethernet im Auto auf den Weg zu bringen. Gleichzeitig wurde Wert auf faire Lizenzierungsbedingungen seitens des Rechteinhabers Broadcom gelegt.

Bereits im März 2012 gab es parallel dazu einen Call for Interest (CFI) zur Gründung einer IEEE-Arbeitsgruppe, die einen Gigabit-Standard für den Einsatz im Automobil entwickeln sollte, mit dem Arbeitsnamen *Reduced Twisted Pair Gigabit Ethernet* (RTPGE) [38]. Gründe für den Bedarf nach so hohen Datenraten (während 100 Mbit/s im Automobil gerade eben erst die ersten Schritte machte) waren Anwendungsfälle wie Kamerasysteme, bei denen 100 Mbit/s nicht für unkomprimierte Übertragung ausreichten [38], oder die Aggregation des Datenverkehrs im Backbone [37]. Wegen langer Entwicklungszyklen in der Industrie wurde frühzeitig mit der Arbeit an höheren Datenraten begonnen, da deren Bedarf absehbar war [54]. Gegen Ende 2012 wurde die Taskforce 802.3bp gegründet, und nahm die Arbeit an dem neuen Standard auf. Anfang 2014 wurde RTPGE umbenannt und bekam den offiziellen Namen IEEE 802.3bp 1000BASE-T1. Zum Zeitpunkt dieser Ausarbeitung (Juli 2016) befand sich der Standard IEEE 802.3bw in der letzten Phase vor Veröffentlichung des *Draft Standard* [39].

Parallel wurde im Frühjahr 2014 entschieden, auch OPEN Alliance *BroadR-Reach* in einen IEEE-Standard umzuwandeln. Die Spezifikation der Bitübertragungsschicht der *BroadR-Reach*-Technologie ist seitdem unter [19] öffentlich verfügbar. Man versprach sich daraus Vorteile wie gemeinsame Entwicklung neuer Features für beide Standards sowie die Koexistenz und die Ausnutzung von Synergieeffekten beider Technologien, wie der CFI von März 2014 aufführt [37]. Im September 2014 fand das erste Arbeitstreffen der auf den CFI hin gegründeten Taskforce 802.3bw statt, und *BroadR-Reach* bekam den Namen IEEE 802.3bw 100BASE-T1. 2015 wurde der Standard zu 100BASE-T1 veröffentlicht [43].

Beide Ethernet-Standards, 100BASE-T1 und 1000BASE-T1 werden auf mittelfristige Sicht als die vorherrschende Netzwerktechnologie im Automobil gesehen. Das Gigabit-Netzwerk wird zur Anbindung von Kameras, zur Realisierung des Backbone-Netzes, zur Diagnose und zur Übertragung von Infotainment-Videos genutzt werden. 100BASE-T1 mit 100 Mbit/s wird seine Anwendung in der Steuergerätekommunikation, der Anbindung von Radarsensoren und Fahrassistenzsystemen und der Vernetzung von Unterhaltungs-, Informations- und Konnektivitätssystemen finden, aber auch klassische Bereiche wie Bremssysteme, Motor- und Getriebesteuerung und Sensoren werden über Ethernet-Links kommunizieren [37].

2.1.3 AUTomotive Open System ARchitecture (AUTOSAR)

Übersicht

Die zunehmende Komplexität von Fahrzeugsystemen und -architekturen führt verstärkt dazu, dass der bisherige, Komponenten-getriebene Entwicklungsprozess durch einen funktionsorientierten abgelöst wird [4]. Die Komponenten, welche ein System realisieren können von verschiedenen Zulieferern kommen. Da vieles Hersteller- oder Fahrzeugspezifisch entwickelt wurde, ist nur ein geringes Maß an Wiederverwendbarkeit gegeben [70]. Aus diesen Gründen wurde 2003 die Entwicklungspartnerschaft AUTOSAR ins Leben gerufen. Das AUTOSAR-Konsortium besteht aus Partnern aus der Automobil- und Zuliefererindustrie. Die acht *Core-Partners* sind BMW, Bosch, Continental, Daimler, Ford, General Motors, PSA Peugeot Citroen, Toyota und Volkswagen. Daneben gibt es über hundert *Premium-*, *Associate-* und *Development-Partners* [4]. Das gemeinsame Ziel ist ein industrieweiter Standard für Systemfunktionen und Schnittstellen im Automobil. Dadurch sollen unter anderem die Austauschbarkeit und die Wiederverwendbarkeit von Softwarekomponenten verbessert und die damit verbundenen Entwicklungskosten reduziert werden. Das Motto der Konsortialpartner, die teilweise in Konkurrenz zueinanderstehen, lautet dabei: *Cooperate on standards, compete on implementation* [4].

Geschichte

2003 nahm die AUTOSAR-Partnerschaft ihre Arbeit auf und begann mit der Ausarbeitung des ersten, 2006 veröffentlichten Standards AUTOSAR Release 2.0. Betriebssystem und Kommunikationsstapel wurden dabei aus dem in ISO-Norm 17356-5 standardisierten OSEK [46] (Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug) übernommen. OSEK wurde 1993 ebenfalls als Standardisierungsgremium mit ähnlicher Besetzung gegründet und arbeitete einen Standard für ein echtzeitfähiges Betriebssystem nebst Kommunikationsstack aus. OSEK ist in gewissem Maße als Vorgänger des AUTOSAR-Projektes zu sehen und lebt in diesem fort. Bis heute stellt OSEK das zugrundeliegende Betriebssystem im AUTOSAR-Standard dar. 2007 wurde AUTOSAR Release 3.0 veröffentlicht und brachte unter anderem ein über alle Bussysteme harmonisiertes Konzept zum Wecken von Steuergeräten und damit dem Start des Netzwerks mit sich. AUTOSAR Release 3.2 (2011) führte erstmalig Teilnetzbetrieb in die Architektur ein, der aktiv auf dem CAN-Bus umsetzbar ist.

Parallel dazu brachte Release 4.x (ab 2009) die Unterstützung von Ethernet und IP im Kommunikationsstack mit sich.

Technischer Überblick

Das Konzept von AUTOSAR baut auf einem Schichtenmodell, der *Layered Architecture* auf, die in [11] detailliert beschrieben ist. Die *AUTOSAR Basic Software* (BSW) stellt das Grundgerüst dar. Sie bietet funktionsunabhängige, infrastrukturelle Dienste, die die Basis für die darüberliegenden Anwendungen (Softwarekomponenten) darstellen und gleichzeitig abstrahiert sie die zugrundeliegende Hardware.

Die Aufgaben der BSW beinhalten unter anderem:

- Abstraktion der Steuergerätehardware
- Abstraktion des Microcontrollers und der Peripherie
- Echtzeitfähiges Multitasking-Betriebssystem
- Zugriff auf Kommunikationsschnittstellen
- Netzwerkmanagement

Über der BSW liegt die AUTOSAR *Runtime Environment* (RTE). Sie stellt eine statisch konfigurierbare Middleware dar. Über sie können Steuergeräte-Funktionen (AUTOSAR-Softwarekomponenten) miteinander kommunizieren, losgelöst von dem tatsächlichen Signalpfad. Durch dieses Modell lassen sich Funktionen als Softwarekomponenten unabhängig von Steuergeräten oder Steuergerätetopologien herstellerübergreifend entwickeln und ohne Änderung in eine Architektur integrieren oder auch wiederverwenden.

2.2 Energiebedarf der E/E-Architektur

Im vorangegangenen Abschnitt wurde verdeutlicht, welche Komplexität die Steuergerätearchitektur in heutigen Fahrzeugen besitzt. Von der Struktur her ist dieses System an verteilten und vernetzten Rechnern durchaus mit dem Datennetzwerk eines kleinen Unternehmens vergleichbar. In Europa beträgt die durchschnittliche Anzahl an Steuergeräten über alle Fahrzeugtypen hinweg heute bereits 45, 2019 wird diese Zahl bereits auf 55 prognostiziert [54]. Dieser Schnitt wird durch Fahrzeuge der Kompaktklasse

und durch niedrige Ausstattungsvarianten stark gedrückt. Im Premiumsegment gibt es heute bereits Fahrzeuge mit über 80 Steuergeräten an Bord. Im Folgenden soll eine Abschätzung gegeben werden, welchen elektrischen Leistungsbedarf die Gesamtheit aller Steuergeräte im Bordnetz verursacht, um ein Gefühl für deren Einfluss auf Verbrauch und Schadstoffemission zu vermitteln.

2.2.1 Klassifikation von Steuergeräten

Steuergeräte unterscheiden sich hinsichtlich Leistungsfähigkeit und Energiebedarf abhängig von ihrem Einsatzzweck. Zur Abschätzung wird eine Klassifikation der Steuergeräte in drei Gruppen anhand der Leistungsaufnahme (hoch (H), mittel (M), gering (G)) vorgenommen.

Für jede Klasse werden die folgenden vier wesentlichen Beiträge zum Leistungsbedarf berücksichtigt:

- *Microcontroller:* Die typischerweise eingesetzten Microcontroller und deren Stromaufnahme stammen aus [73]. Viele Steuergeräte setzen mittlerweile stärkere Prozessoren, teilweise sogar zwei gleichzeitig ein. deswegen wurde das 1,5-fache der Stromaufnahme in den Klassen zugrunde gelegt.
- *Peripherie:* Hierzu gehören Speicherbausteine, externe Controller und Logikbausteine. Für Klassen H, M und G werden pauschal 2 W, 1 W und 0,5 W angenommen.
- *Netzwerkinterfaces:* Abhängig von der Komplexität besitzen Steuergeräte eine unterschiedliche Zahl unterschiedlicher Netzwerkschnittstellen, wie beispielsweise CAN, FlexRay, MOST oder Ethernet. Die Leistungsaufnahme pro Port wurde in Kapitel 3 analysiert. Für Geräte der Klasse H wurden 500 mW angesetzt, entsprechend einer Ethernet-Schnittstelle oder zweier FlexRays. Für Klasse M wurden 120 mW als Mittelwert zwischen einem FlaxRay und einem CAN-Interface veranschlagt. Für Klasse G wurden 40 mW für ein CAN-Interface verwendet.
- *Wirkungsgrad:* Die komponentenseitige Leistungsaufnahme verursacht eine entsprechend dem Wirkungsgrad der Netzteile höhere bordnetzseitige Leistungsaufnahme des Steuergerätes. Linearregler weisen einen geringen Wirkungsgrad auf, während Schaltregler sehr hohe Wirkungsgrade (80-90 %) besitzen. Da beide Reglertypen genutzt werden, lässt sich hier nicht differenzieren, so dass im Mittel ein Wirkungsgrad von 50 % angenommen wird.

Tabelle 2.1: Klassifikation von Steuergeräten nach geschätzter Leistungsaufnahme

Leistungsklasse	Microcontroller	Peripherie	Netzwerk	Wirkungsgrad	
Hoch	2.5 W	2 W	0.5 W	50 %	10 W
Mittel	0.8 W	1 W	0.12 W	50 %	4 W
Gering	0.5 W	0.5 W	0.04 W	50 %	2 W

Damit ergibt sich Tabelle 2.1. Um den Leistungsbedarf einer Steuergerätearchitektur abzuschätzen, können alle Steuergeräte gemäß dieser Klassifikation eingeordnet werden. Aus Gründen der Vertraulichkeit können an dieser Stelle keine Details einzelner Architekturen veröffentlicht werden. Deswegen wird im Folgenden nur das Ergebnis der Abschätzungen gezeigt.

2.2.2 Abschätzung der Leistungsaufnahme

Eine Klassifikation der Architektur eines Fahrzeugs aus dem Premiumsegment ergab eine Aufteilung von 25 H-Steuergeräten, 23 M-Steuergeräten und 51 G-Steuergeräten, bei einer Gesamtanzahl von 99. Dementsprechend ergibt sich eine gesamte Leistungsaufnahme von etwa 450 Watt.

Ein Architekturbeispiel von Schmutzler *et al.* [72] zeigt ein Mittelklassefahrzeug deren Klassifikation 17 H-Steuergeräte und 25 M-Steuergeräte ergibt, was einen Leistungsbedarf von etwa 270 Watt ergibt. G-Steuergeräte wurden in diesem Beispiel nicht aufgeführt. Geht man vereinfachend von einer Gleichverteilung aller Klassen aus, ergibt sich der in Abbildung 2.1 gezeigte Verlauf der Leistungsaufnahme. Die beiden analysierten, konkreten Beispielarchitekturen sind ebenfalls markiert. Der Fehler durch die Annahme der Gleichverteilung ist lediglich gering.

Diese Zahlen stellen den heutigen Zustand dar. In Zukunft wird der Leistungsbedarf der Bordelektronik weiter ansteigen. Vernetzung mit der Umwelt, Fahrassistenzsysteme und nicht zuletzt das automatisierte Fahren führen zu einer weiteren Zunahme an Steuergeräten und zu höheren Anforderungen – und damit zu höherer Leistungsaufnahme – aller Komponenten (insbesondere Prozessoren, Sensoren und Netzwerkchips). Eine Größenordnung ist heute noch schwer abschätzbar, da sich in diesem Bereich derzeit viel bewegt.

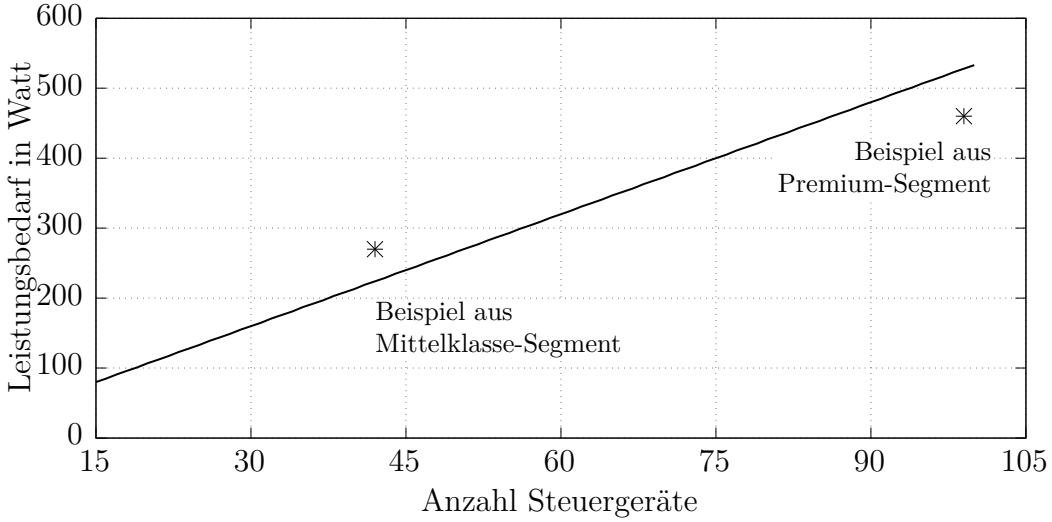


Abbildung 2.1: Abgeschätzte Gesamteistungsaufnahme bei einer Gleichverteilung von H-, M-, und G-Steuergeräten

2.2.3 Einfluss auf Schadstoffemission und Verbrauch

Mit der Näherung aus dem letzten Abschnitt und den Äquivalentumrechnungen aus Kapitel 1.2.2 lässt sich der Einfluss der Bordelektronik auf Schadstoffemission und Verbrauch abbilden. Bei einem Leistungsbedarf von 250 Watt bis 600 Watt verursacht die reine Steuergeräte-Elektronik, ohne Aktoren und sonstige Verbraucher also einen Mehrverbrauch von $0,4$ l / 100 km bis 1 l / 100 km, beziehungsweise 9 g/km CO_2 bis 21 g/km CO_2 an Emissionen.

2.2.4 Senkung des Leistungsbedarfs

Wie bereits erwähnt, ist davon auszugehen, dass der Leistungsbedarf der Elektronik mit neuen Innovationen und damit höheren Anforderungen an die Hardware weiter zunehmen wird. Auch die Einführung von Ethernet als Vernetzungstechnologie steigert den Energiehunger. Gleichzeitig ist aber auch der Druck auf die Industrie groß, Verbrauch und Kohlenstoffdioxidemissionen zu reduzieren. Beides miteinander in Einklang zu bringen, ist nur möglich, wenn der Energiebedarf neuer Innovationen und Technologien durch geschickte Maßnahmen reduziert und so dem stetigen Anstieg des Strombedarfs entgegengewirkt wird. Zum einen können die Komponenten selbst optimiert werden. Neben der Verbesserung von Microcontrollern, Spannungsreglern und Peripheriebausteinen durch neue Halbleitertechnologien sind auch die Vernetzungstechnologien ein

wichtiger Ansatzpunkt. Im folgenden Kapitel wird der Leistungsbedarf von Ethernet betrachtet, dem von heute eingesetzten Bustechnologien gegenübergestellt und *Energy Efficient Ethernet*, eine Erweiterung des Ethernet-Standards, als Möglichkeit zur Senkung der Stromaufnahme von Ethernet-Transceivern untersucht. Zum anderen kann Energie im Bordnetz durch geschicktes Netzwerk- und Energiemanagement eingespart werden, indem während des Betriebs nicht benötigte Steuergeräte abgeschaltet und bei Bedarf wieder aufgeweckt werden. Dies wird als Teilnetzbetrieb bezeichnet und stellt einen Schwerpunkt der vorliegenden Arbeit dar. Im späteren Verlauf wird ein Konzept vorgestellt, mit dem Teilnetzbetrieb bei Ethernet-basierten Fahrzeugnetzen umgesetzt werden kann.

3 Energy Efficient Ethernet im Bordnetz

3.1 Energieeffizientes Ethernet

Hintergrund

Green IT bezeichnet eine in den 90er Jahren entstandene Bewegung, auch IT-Geräte energieeffizienter zu gestalten. Eines der bekannten Beispiele aus dieser Bewegung ist der *Energy Star* der amerikanischen *Environmental Protection Agency*, der Geräten stromsparendes Verhalten wie das automatische Wechseln in den Standby-Zustand nach einer Zeit der Nichtbenutzung bescheinigt.

Auch die weltweiten Datennetze, allen voran das Internet, wurden mit einem Auge auf deren Energieeffizienz betrachtet [35]. Netzwerktechnologien wie Ethernet wurden ursprünglich nicht mit der Zielsetzung einer besonders ausgeprägten Energieeffizienz entwickelt. Technologische Machbarkeit, Performance und geringe Kosten standen im Vordergrund, da auch die Verbreitung von informationstechnischen Geräten in den 80er- und 90er-Jahren noch nicht die Ausmaße annahm, dass deren Energiebedarf einen bedeutenden Posten auf der globalen Stromrechnung ausgemacht hätte. Dies sollte sich jedoch ändern. Im Jahr 2000 verursachten allein in den Vereinigten Staaten die Ethernet-Schnittstellen aller Endgeräte, Switches und Router etwas über sechs Terawattstunden an Stromverbrauch [35]. Man sah ein rasantes, globales Wachstum von Netzwerken und dem Internet, weswegen Ideen und Möglichkeiten gesucht wurden, um auch an dieser Stelle den Energieverbrauch zu reduzieren. Im Zuge dieser Bewegung fand eine Idee ihren Ursprung, Netzwerkkarten in einen Stromsparmodus zu versetzen, während sie keine Pakete zu übertragen hatten [35]. Diese Idee manifestierte sich in IEEE 802.3az *Energy Efficient Ethernet* (EEE), einer Erweiterung des Ethernet-Standards, die mittlerweile in den Standard eingeflossen ist [40]. Nach ersten

Abschätzungen ließen sich die weltweiten, durch Stromverbrauch bedingten jährlichen Ausgaben um eine Milliarde US-Dollar senken [24].

Technologie

Je nach Auslastung des Netzwerks werden auf einem Ethernet-Link nicht ständig Frames übertragen, es gibt auch Zeiten zu denen aus Sicht des Datenverkehrs Ruhe auf einer Verbindung herrscht. Auf der Bitübertragungsschicht (ab 100BASE-TX) manifestiert sich diese Ruhe jedoch durch eigens dafür vorgesehene Leitungscodes (*Idle*-Codes). Diese zu übertragen kostet genau so viel Energie, wie die Übertragung von Daten-Codes. Auf den Energiebedarf bezogen macht es daher keinen Unterschied, ob gerade Frames übertragen werden oder nicht, also ob die Netzwerkauslastung hoch ist oder gering. EEE soll die Leistungsaufnahme des Ethernet-PHYs (*Physical Layer Transceiver*, die Bitübertragungsschicht von Ethernet) abhängig von der Netzwerkauslastung senken, indem während der *Idle*-Zeiten auch auf der Bitübertragungsschicht innerhalb gewisser Grenzen Ruhe herrscht. Das Funktionsprinzip von EEE ist einfach und beschränkt sich auf die Bitübertragungsschicht von Ethernet. Während keine Daten übertragen werden, wird der Ethernet-PHY in den *Low Power Idle*-Zustand (LPI) versetzt, anstatt *Idle*-Codes zu senden. In diesem Zustand wird die Leistungsaufnahme des PHYs um bis zu 90 % reduziert [24]. Dadurch wird Energie gespart, während keine Pakete übertragen werden. Der Übergang vom Normal- in den LPI-Betrieb benötigt eine gewisse Zeit T_S, die gemäß IEEE 802.3 für 100BASE-TX bei 200 µs liegt [40]. Steht ein Paket zur Übertragung an, muss der PHY zunächst wieder in den Normalbetrieb wechseln, was die Zeit T_W dauert, die im Standard für 100BASE-TX 30 µs beträgt [40]. Um die Synchronisation zwischen Sender und Empfänger aufrecht zu erhalten (dies geschieht durch Taktrückgewinnung aus dem Leitungssignal), wird nach der (vergleichsweise hohen) Zeitdauer T_Q (24 ms bei 100BASE-TX [40]) eine kurze *Refresh*-Phase eingelegt, bei der der PHY kurze *Refresh*-Pulse sendet, um die Verbindung aufrecht zu erhalten. Da das Senden der *Refresh*-Pulse verglichen mit T_Q sehr kurz ist, fallen die *Refresh*-Phasen bei der Betrachtung der Leistungsaufnahme nicht ins Gewicht und werden deswegen in den folgenden Abschnitten ignoriert.

Die Übergangszeiten T_S und T_W sind jedoch von großer Bedeutung, da der PHY während dieser Zeiten eine hohe Leistungsaufnahme (bis zu 100 % der Leitungsaufnahme im Normalbetrieb [24]) aufweist, aber keine Pakete übertragen werden. Dies wird an späterer Stelle genauer betrachtet.

EEE im Automobil

EEE hat seinen Ursprung in der Welt der Büro- und Heimvernetzung, und reduziert den Energiebedarf der Ethernet-Schnittstellen in Phasen geringer Auslastung, die insbesondere in kleinen Heim- und Büronetzwerken überwiegen. Ob sich der Einsatz von EEE bei der Vernetzung eingebetteter Steuergeräte im Fahrzeug lohnt, hängt stark von der Charakteristik des Datenverkehrs auf den Netzwerklinks ab, wie später gezeigt wird. Dass Energieeffizienz im Bordnetz eine wichtige Rolle spielt, wurde bereits dargestellt, und Ethernet bringt zunächst einmal – verglichen mit klassischen Bussystemen – einen hohen Leistungsbedarf mit sich.

Um ein Gefühl für die Größenordnungen zu vermitteln, wird im Folgenden eine kurze Einordnung vorgenommen, indem zunächst der Leistungsbedarf heute im Fahrzeug eingesetzter Bustechnologien analysiert wird und dem Leistungsbedarf von 100BASE-TX Ethernet gegenübergestellt wird. Dabei darf nicht außer Acht gelassen werden, dass Ethernet durch seine vielfach höhere Datenrate und Skalierbarkeit viele neue Funktionen im Automobil überhaupt erst ermöglicht, und das Plus an Leistungsbedarf somit auch mit einem Zugewinn an Funktionalität einhergeht. Anschließend wird der Einfluss von EEE untersucht und gezeigt, unter welchen Bedingungen die Zusatzfunktion nennenswerte Einsparungen ermöglicht. Die im Folgenden dargestellten Untersuchungen und Ergebnisse wurden auch in [94] veröffentlicht und durch Kunze *et al.* weiterverwendet.

3.2 Energiebedarf klassischer Bussysteme

Die heute im Fahrzeug verwendeten Bussysteme wurden bereits in Abschnitt 2.1 vorgestellt. Im Folgenden wird der Strombedarf der Transceiver der einzelnen Bustechnologien modelliert, um eine Vergleichbarkeit zu Ethernet und zu *Energy Efficient Ethernet* zu schaffen.

3.2.1 Controller Area Network

Da CAN ein nicht-synchrones Bussystem mit geteiltem Medium ist, treten Kollisionen auf, die mittels CSMA/CA aufgelöst werden. Ab einer Buslast von 60 % kommt es statistisch gesehen so oft zu Kollisionen, dass der Datendurchsatz einbricht. Es ist

daher gängige Praxis, CAN-Busse nur mit einer Last von 40 % zu betreiben, um einen gewissen Puffer zu behalten [34].

Ein CAN-Transceiver benötigt während des Sendens eines Rahmens die Leistung P_{tx} und die vergleichsweise niedrige Leistung P_{rx} sonst (d.h. während er empfangsbereit ist oder gerade empfängt). Die Leistung während des Sendens P_{tx} kann angenähert werden als Mittelwert der Leistungsaufnahme P_{dom}, die zum Treiben eines dominanten Buszustandes notwendig ist, und der Leistungsaufnahme P_{rec}, die anfällt wenn der Transceiver die Busleitung in rezessivem Zustand belässt. Dabei wird angenommen, dass im Frame im Schnitt gleich viele dominante wie rezessive Bits vorkommen.

$$P_{tx} = \frac{1}{2}(P_{dom} + P_{rec}) \tag{3.1}$$

P_{dom} und P_{rec} sind Werte, die in den Datenblättern gängiger CAN-Transceiver angegeben sind. Da stets nur ein Knoten einen Rahmen sendet, während alle anderen empfangen, kann man die gesamte Leistungsaufnahme aller CAN-Transceiver P_{CAN} wie folgt abschätzen.

$$P_{CAN} = uP_{tx} + (n - u)P_{rx} \tag{3.2}$$

Dabei bezeichnet u die Buslast und n die Anzahl der CAN-Knoten auf dem Bus. Die Werte für P_{dom} und P_{rec} wurden aus den Datenblättern mehrerer CAN-Transceiver[1] ermittelt. Dabei ergab sich P_{tx} zu 140 mW und P_{rx} zu 35 mW.

3.2.2 FlexRay

FlexRay ist, wie CAN, ein Bussystem mit geteiltem Medium, daher gelten die selben grundlegenden Annahmen. Ein Unterschied besteht darin, dass eine Auslastung von 100 % möglich ist, da FlexRay zeitsynchron arbeitet. Ein weiterer Unterschied zu CAN ist, dass es bei FlexRay sogenannte *Active Stars* (aktive Sternverteiler) gibt. *Active Stars* verteilen und verstärken das Bussignal auf mehrere Zweige [69]. Dies birgt mehrere Vorteile, allen voran eine Verbesserung von Abstrahlung und Störfestigkeit [67] sowie höhere Fehlertoleranz (ein elektrischer Fehler auf einem Zweig beeinflusst nicht den Rest des Busses). Wie auch bei CAN sendet immer nur ein FlexRay-Knoten einen Frame, während alle anderen empfangen. Ein *Active Star* jedoch verteilt das Signal auf alle seine Zweige, so dass zusätzlich zu dem Bus-Treiber des Senders noch die der Sterne aktiv senden (und erhöhte Leistungsaufnahme verursachen).

[1] NXP TJA1040, Atmel ATA6660, Maxim MAX13041

k_1 sei die Anzahl gleichzeitig sendender Bus-Transceiver, k_0 die Anzahl der empfangenden Transceiver, n die Anzahl der Knoten, s die Anzahl der *Active Stars* auf dem Bus, und b_i die Anzahl Zweige des *Active Star i*. Dann ist die Anzahl $k1$ der gleichzeitig aktiven Bustreiber die Summe aller Äste aller *Active Stars* abzüglich jeweils des Astes von dem gerade empfangen wird, plus Transceiver des ursprünglichen Senders.

$$k_1 = \left[1 + \left(\sum_{i=0}^{s} b_i\right) - s\right] \tag{3.3}$$

Die Anzahl k_0 der empfangenden Transceiver entspricht der Anzahl der Knoten abzüglich des ursprünglichen Senders (alle anderen Knoten haben genau einen empfangenden Transceiver, auch die *Active Stars*).

$$k_0 = n - 1 \tag{3.4}$$

Die Leistungsaufnahme aller FlexRay-Transceiver P_{Flx} lässt sich dann als Funktion von k_1 und k_0, der Leistungsaufnahme während des Sendens P_{tx} und während des Empfangens P_{rx} und der Buslast u modellieren.

$$P_{Flx} = k_1 u P_{tx} + ((1-u)k_1 + k_0)P_{rx} \tag{3.5}$$

Ist $k_1 = 1$ ergibt sich die selbe Gleichung wie für CAN (Gleichung 3.2), da in diesem Falle keine *Active Stars* vorhanden wären, die den einzigen topologischen Unterschied darstellen.

Auch hier wurden gemittelte Datenblattwerte verschiedener FlexRay-Transceiver[2] als Modellparameter verwendet. Dabei ergab sich P_{tx} zu 200 mW und P_{rx} zu 80 mW.

3.2.3 Media Oriented Systems Transport

Wie bereits beschrieben, ist MOST ein synchrones Ringnetzwerk. Ein Frame wird immer initiiert durch den *Timing Master* und an dessen direkten Nachbarn in der festgelegten Richtung gesendet. Jeder Knoten leitet wiederum die empfangenen Bits an jeweils seinen eigenen Nachbarn weiter, so dass der *Timing Master* nach der Ausbreitungsverzögerung die von ihm gesendeten Bits wieder selbst empfängt. Der Ring ist damit geschlossen. Durch die zeitsynchrone Natur von MOST weiß jeder Knoten, in

[2] NXP TJA1080, TJA1081 Austria Microsystems AS8221, ELMOS E910.54

welcher Zeitscheibe er Daten senden soll und fügt zur entsprechenden Zeit selbst Bits in den Frame ein. Da jeder Knoten stets wiederholen muss, was er gerade empfängt, benötigt jedes MOST-Interface am Bus stets volle Sendeleistung und die gesamte Leistungsaufnahme ist unabhängig von der genutzten Datenrate und nur von der Anzahl Knoten n. Eine passende Metapher zu MOST ist in dieser Hinsicht ein Zug, der stets im Kreis fährt, egal ob mehr oder weniger Passagiere darinsitzen.

Das MOST-Interface besteht aus dem sogenannten *Fiber Optical Transceiver* (FOT), der im Wesentlichen aus einer Leuchtdiode als Sender und einer Photodiode oder einem Phototransistor als Empfänger besteht sowie dem *Intelligent Network Interface Controller* (INIC), in dem der Rest der Bitübertragungsschicht von MOST und dessen höhere Schichten implementiert sind. Die Leistungsaufnahme aller MOST-Transceiver lässt sich demnach einfach als die Summe aller FOTs und INICs am Bus ausdrücken.

$$P_{MOST} = n \cdot (P_{FOT} + P_{INIC}) \tag{3.6}$$

Die in den jeweiligen Datenblättern angegebene Leistungsaufnahme fünf verschiedener FOTs[3] wurde gemittelt und als Modellparameter herangezogen. Da es nur einen Hersteller des INIC gibt (SMSC, [76]), wurde der Datenblattwert des OS81050 INIC verwendet. P_{FOT} beträgt damit 286 mW und P_{INIC} 479 mW.

3.3 Energiebedarf von Ethernet

Bei modernen Varianten von Ethernet (100BASE-TX, 100BASE-T1, 1000BASE-T, u.a.) ist die Leistungsaufnahme des Ethernet-Transceivers unabhängig von der Auslastung (bzw. genutzten Datenrate), sofern IEEE802.3az *Energy Efficient Ethernet* nicht genutzt wird. Werden keine Ethernet-Frames auf dem Link übertragen, werden in der Zwischenzeit *Idle*-Codegruppen auf dem Medium gesendet [40]. Dadurch wird eine kontinuierliche Taktsynchronisation und Lebenderkennung des Links gewährleistet. Das Übertragen von *Idle*-Codegruppen benötigt jedoch genau so viel Leistung, wie das Übertragen jeglicher anderer Codes. Demnach ist auch bei Ethernet die gesamte Leistungsaufnahme hauptsächlich von der Anzahl Ports (also Ethernet-Transceiver) im Netz abhängig. Jedoch spielt dabei auch die Topologie noch eine Rolle. Ethernet ist ein geswitchtes Netzwerk, und die Switches selbst benötigen ebenfalls elektrische

[3] Hamamatsu L/S10663/4, L/S10063/4, Firecomms FCM110R/D, Avago AFBR-1012/2010, AFBR-1150L/2150L

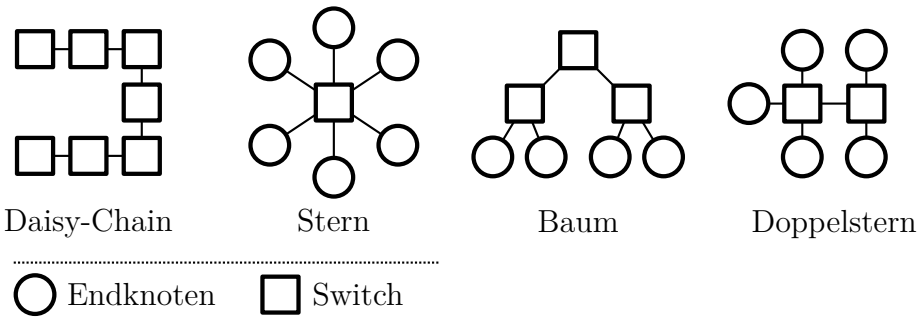


Abbildung 3.1: Grundsätzliche Topologieformen

Leistung. Allgemein gilt, je mehr Switches sich in der Topologie befinden, desto höher ist die gesamte Leistungsaufnahme.

Für ein schleifenfreies Punkt-zu-Punkt-Netz wie Ethernet ist die Anzahl aktiver Ports k (und damit Transceiver) unabhängig von der Topologie (und Anzahl Switches), sondern nur von der Gesamtanzahl Knoten n, von denen jeder entweder ein Endknoten e oder ein Switch s sein kann.

$$k = 2(e + s - 1) = 2(n - 1) \tag{3.7}$$

Betrachtet man die Beispieltopologien in Abbildung 3.1, sieht man, dass die Anzahl an Netzwerkports für alle Varianten gleich ist, unabhängig von der Verteilung von s und e, solange $s + e = n$ konstant ist. Der topologieabhängige Anteil am gesamten Leistungsbedarf wird rein durch die Switches selbst (abzüglich deren Transceiver, die ja bereits berücksichtigt sind) verursacht. Die messbare und in den Datenblättern angegebene Leistungsaufnahme P_{SW} eines Switches beinhaltet einen konstanten Anteil P_0 (Switch-Backplane, Speicher, Peripherie), sowie die Leistungsaufnahme seiner p Ports.

$$P_{SW} = p \cdot P_{Port} + P_0 \tag{3.8}$$

P_{Port} wird maßgeblich durch die PHYs verursacht, die bereits berücksichtigt wurden. Deswegen lässt sich die Leistungsaufnahme einer Ethernet-Topologie näherungsweise ausdrücken als die Leistungsaufnahme aller k Ports im Netzwerk plus konstanten Anteil P_0 aller s Switches.

$$P_{ETH} = k \cdot P_{Port} + s \cdot P_0 \tag{3.9}$$

Zur Ermittlung des konstanten Anteils P_0, wurden die Datenblätter von zwölf vergleichbaren 100BASE-TX Switches[4] analysiert. Da die Switches über eine unterschiedliche

[4]Micrel KSZ8999, KSZ8997, KSZ8842, KSZ8995, KSZ8873, SMSC LAN98393, Realtek RTL8324, RTL8316, RTL8309, RTL8305, Marvell 88E6061

Anzahl Ports verfügen, ließ sich eine Näherung für die Modellparameter P_0 mit ca. 260 mW und P_{Port} mit ca. 220 mW ermitteln.

Zusätzlich wurden die Datenblätter von neun externen 100BASE-TX PHYs[5] untersucht, wobei sich im Mittel eine Leistungsaufnahme von 230 mW ergab, was sehr dicht beim gewählten Modellparameter P_{Port} liegt.

Tabelle 3.1 zeigt die Auswertung des Modells für die in Abbildung 3.1 gezeigten Ethernet-Topologien für eine jeweils unterschiedliche Anzahl vernetzter Geräte.

Tabelle 3.1: Modellierte Leistungsaufnahme verschiedener Ethernet-Topologien

Knoten	Daisy Chain	Baum	Stern	Doppelstern
5	2,5 W	–	2.0 W	2,3 W
7	3,9 W	3,4 W	2,9 W	3,2 W
10	6,0 W	–	4,2 W	4,5 W
15	9,5 W	8,0 W	6,4 W	6,7 W
31	20,7 W	17,1 W	13,4 W	13,7 W

3.4 Energy Efficient Ethernet

Der PHY – bzw. die Übertragung von Codes auf dem Link – ist für den Großteil der Leistungsaufnahme bei Ethernet verantwortlich. EEE versetzt den Sendeteil des lokalen PHYs und den Empfangsteil des entfernten PHYs in den LPI-Zustand, während keine Datenframes übertragen werden. Anstatt *Idle*-Codes zu übertragen, wie es ohne EEE der Fall wäre, verstummt der Sender. In diesem Fall liegt die Leistungsaufnahme etwa bei 10 % derer im Normalbetrieb [24, 71]. Sobald ein Frame zur Übertragung ansteht, werden sowohl der lokale Sendeteil als auch der entfernte Empfangsteil aus dem LPI-Zustand zurückgeholt, bevor der Frame übertragen wird. Zu Zeiten geringer Linkauslastung lässt sich somit ein prozentual hoher Anteil Energie auf der Verbindung sparen. Rein theoretisch, würde man die Übergangszeiten zwischen LPI- und Normalbetrieb vernachlässigen, ließen sich so auf einer zur Hälfte ausgelasteten Verbindung im Mittel ca. 45 % Energie sparen. Jedoch spielen genannte Übergangszeiten eine große

[5] Marvell 88E3016, National DP83848K, Micrel KSZ8041NL, KSZ8051MLL, KSZ8031RNL, Broadcom BCM5238, BCM5241, SMSC LAN8710A, LAN88710AM

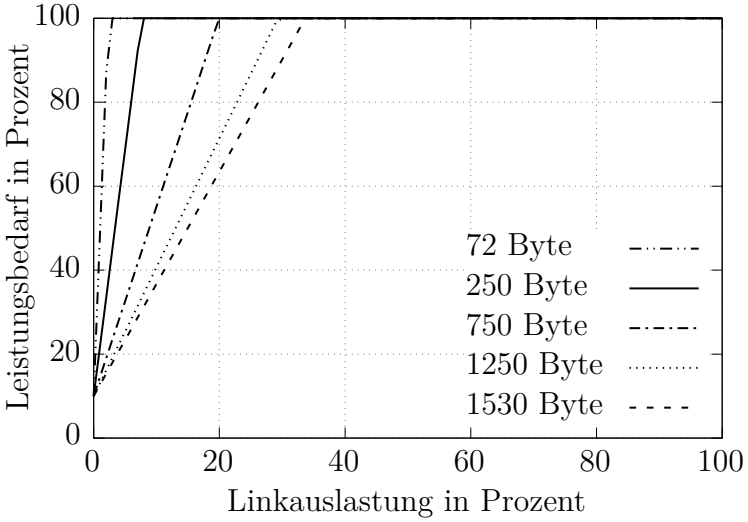


Abbildung 3.2: Einfluss von Framegröße und Auslastung auf EEE bei zyklischer Datenübertragung

Rolle. Sie sind jeweils abhängig vom Physical Layer in IEEE 802.3 verankert. Für 100BASE-TX beträgt T_s 200 µs und T_w 30 µs, wie bereits weiter oben erwähnt. Bei einer Datenrate von 100 Mbit/s dauert die Übertragung eines 1500 Byte großen Frames 120 µs, also weniger lange als die Übergangszeit. Während der Übergangszeiten weist der PHY zwar seine volle Leistungsaufnahme auf [24], kann jedoch keinen Frame übertragen. Das reduziert die Wirksamkeit von EEE teilweise deutlich in Abhängigkeit von Framegröße und der zeitlichen Verteilung der Frames, wie von Reviriego *et al.* [71] dargestellt. Insbesondere bei kurzen Frames, die häufig und zeitlich gleichmäßig verteilt auftreten, wirkt EEE kaum, da die Übergangszeiten im Vergleich zur mit dem Senden von Frames verbrachten Zeit sehr groß sind. Dies beschreibt in etwa die Charakteristik, die auf vielen Verbindungen im Fahrzeugnetzwerk zu erwarten ist und die geprägt ist von zyklischem Datenverkehr wie Sensorwerten, Audio- und Videoströmen oder Steuerungsdaten.

Eine Betrachtung mit zyklischem Auftreten von Paketen mit Variation der Paketgröße ergibt einen linearen Zusammenhang zwischen eingesparter Energie und der Link-Auslastung (vgl. Abbildung 3.2). Mit zunehmender Auslastung (Framegröße oder -frequenz) sinkt die Pause zwischen zwei Frames bis zu dem Punkt, an dem sie kürzer ist als die Übergangszeiten. Ab dieser Auslastung erreicht der Frame den LPI-Zustand überhaupt nicht mehr und benötigt stets die volle Leistung.

Diese Grenzauslastung u_g ist abhängig von der Framegröße N.

$$u_g = \frac{N}{\left(\left(T_s + T_w + \left(\frac{N}{R}\right)\right) \cdot R\right)} \tag{3.10}$$

R bezeichnet dabei die Datenrate des Links (100 Mbit/s bei 100BASE-TX). Die prozentuale Leistungsaufnahme unterhalb des Grenzpunktes lässt sich linear aus dem Leistungsbedarf bei Normalbetrieb P_{norm} und LPI P_{LPI} und der Auslastung u berechnen.

$$P_{EEE} = \frac{P_{norm} - P_{LPI}}{u_g} \cdot u + P_{LPI},\ u < u_g \tag{3.11}$$

Je kleiner die Framegröße ist, desto niedriger liegt die Grenzauslastung ab welcher EEE wirksam ist. Bei gleicher Auslastung wird also bei kleinerer Framegröße mehr Leistung benötigt.

Ethernet-Links sind jedoch Vollduplex-Verbindungen, demnach arbeitet auch EEE auf einem Link für beide Richtungen unabhängig. An dieser Stelle kommt die Symmetrie zwischen Upstream und Downstream ins Spiel. Während etwa Daten gesendet aber zeitgleich keine empfangen werden, ist der Sendeteil des lokalen PHYs im Normalbetrieb und der Empfangsteil im LPI, während es beim entfernten PHY gerade andersherum ist. Auf den Link bezogen wird also bei asymmetrischer Auslastung dennoch Energie gespart. Um dies zu beschreiben, ist eine geeignete Maßzahl für die Symmetrie notwendig, die im Folgenden als Symmetriefaktor α bezeichnet wird. Er hängt ab von Upstream-Auslastung u_{up} und Downstream-Auslastung u_{dn} und ist wie folgt definiert.

$$\alpha = \left|\frac{u_{up} - u_{dn}}{u_{up} + u_{dn}}\right| \tag{3.12}$$

Bei symmetrischer Auslastung ist u_{up} gleich u_{dn} und damit $\alpha = 0$. Im anderen Grenzfall, bei vollständig unidirektionaler Auslastung, ist entweder u_{up} oder u_{dn} null, und damit $\alpha = 1$. Der Leistungsbedarf eines EEE-Links kann berechnet werden, indem Formel 3.11 für u_{up} und u_{dn} separat ausgewertet wird, und die Ergebnisse gemittelt werden.

Abbildung 3.3 zeigt den Einfluss der Symmetrie anhand zyklischer, 1250 Byte großes Frames. Die x-Achse zeigt die gemittelte Auslastung beider Richtungen $\frac{1}{2}(u_{up} + u_{dn})$. Es ist zu sehen, dass bei gleicher mittlerer Auslastung EEE wirksamer ist, desto asymmetrischer die Lastverteilung ist.

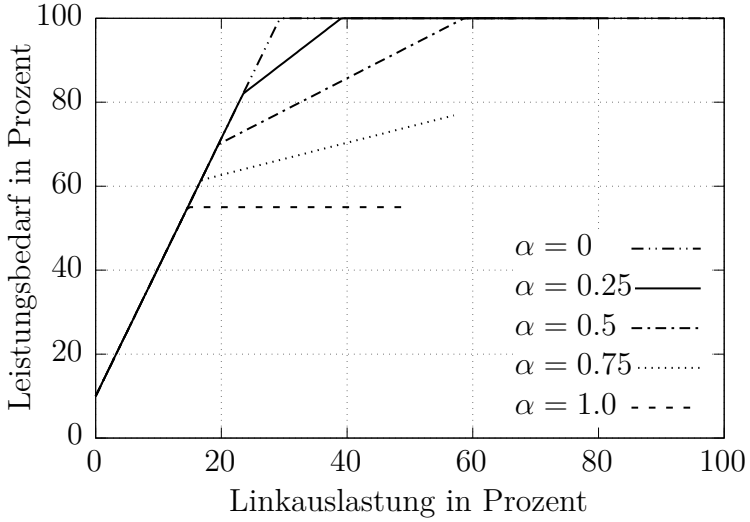


Abbildung 3.3: Einfluss der Lastsymmetrie auf EEE bei zyklischen, 1250 Byte großen Frames

3.5 Lohnt sich Energy Efficient Ethernet im Bordnetz?

Dass Ethernet als Netzwerk ins Automobil kommt, bleibt nicht anzuzweifeln, ebensowenig wie die Tatsache, dass Ethernet, als viel mächtigeres und skalierbareres Netzwerk als die heute üblichen Bussysteme, einen maßgeblich höheren Leistungsbedarf mit sich bringt. Energy Efficient Ethernet ist heute noch nicht Teil der im Fahrzeug einsetzbaren Übertragungstechnologie 100BASE-T1 (*BroadR-Reach*). Es stellt sich die Frage, ob EEE für künftige Ethernet-Varianten in der Fahrzeugvernetzung als Funktion mit aufgenommen werden soll. Dagegen spricht beispielsweise, dass die Übergangsphase zwischen LPI und Normalbetrieb zu gegebenenfalls unerwünschten Verzögerungen bei der Übertragung eines Frames führt. Demnach bedeutet eine Integration von EEE zusätzlichen Aufwand, um solche Einflüsse, insbesondere auch auf Dienstgütemechanismen wie Zeitsynchronisierung, Traffic Shaping und zugesicherte Latenzen (*Time Sensitive Networking*, eine Arbeitsgruppe im Rahmen von IEEE 802.1, die Dienstgüte-Standards für Ethernet erarbeitet [36]) zu untersuchen, und gegebenenfalls sogar Anpassungen am Standard vorzunehmen. Ob dieser Aufwand sich rechnet muss die Industrie entscheiden. An dieser Stelle soll dazu abgeschätzt werden, in welcher Größenordnung der Leistungsbedarf von Ethernet mit und ohne EEE liegt. Wie vorhin gezeigt wurde, ist die Wirksamkeit von EEE stark Fallabhängig und es kann keine allgemeingültige Aussage getroffen werden.

3.5.1 Kriterien für die Wirksamkeit von EEE

Aus dem vorhergehenden Abschnitt können die Bedingungen abgelesen werden, unter denen EEE auf einem Link nennenswerte Energieeinsparungen ermöglicht.

Bei niedriger Auslastung des Links, d.h. bei einer im Vergleich zur Bandbreite geringen Nutzdatenrate, mit der eine Anwendung (bzw. ein Steuergerät) kommuniziert, bringt LPI erhebliche Leistungsersparnisse gegenüber Ethernet. Ist die Linkauslastung beispielsweise nur 5 %, was immer noch einer Datenrate von 5 Mbit/s (oder dem fünffachen der gesamten bei CAN verfügbaren Datenrate) entspricht, reduziert sich der Leistungsbedarf in Senderichtung um fast 80 %, große Frames vorausgesetzt (vgl. Abbildung 3.2).

Bei gleicher Auslastung lässt sich durch EEE mit großen Frames mehr Leistung einsparen als mit kleinen. Bei Anwendungen, die nicht nur kleine Daten-Samples sondern große Pakete mit viel Information und hohen Zykluszeiten übertragen, birgt EEE gegenüber Ethernet viel Potential. Bei einer Auslastung von 20 % reduziert EEE den Leistungsbedarf in Senderichtung um 40 % bei 1530 Byte großen Frames, jedoch bei Frames kleiner als 750 Byte überhaupt nicht (vgl. Abbildung 3.2).

Auf Links mit einer hohen, aber einseitigen Auslastung ist EEE ebenfalls wirksam. Bei vielen Anwendungen, insbesondere dem Streaming von Sensor-, Kamera- oder Audiodaten ist dies der Fall. Die Quelle sendet eine hohe Datenmenge, was zu einer hohen Auslastung des Links in Senderichtung führt, empfängt aber nur Steuerkommandos mit einem Bruchteil der Streaming-Datenrate. In solchen Fällen ist bis zu 45 % Leistungsersparnis auf dem Link realistisch (vgl. Abbildung 3.3).

3.5.2 Beispiele

Der Einfluss von EEE kann demnach nur auf die Charakteristik jedes einzelnen Links bezogen abgeschätzt werden, und nicht pauschal auf eine ganze Topologie, ohne den gesamten Datenverkehr zu kennen. Im Folgenden werden deswegen zwei modellhafte Beispiele gezeigt, die an reale Anwendungsfälle in heutigen Fahrzeugen angelehnt sind, deren Datenverkehr charakterisiert und das Potential von EEE darauf abgeschätzt.

Ethernet im Infotainment

Heutige Mittel- und Oberklassefahrzeuge besitzen eine Vielzahl an Unterhaltungs- und Fahrerinformationssystemen, die in der Regel (noch) als MOST-Ring untereinander vernetzt sind. Ein Beispielnetz könnte einen Digitalradio-Tuner, einen digitalen Verstärker, ein *Consumer Device Gateway* zur Anbindung von Mobiltelefonen oder Tablets der als Medienserver agiert, einen CD-Wechsler und eine Head-Unit (Steuerung) enthalten. Der Leistungsbedarf des zugehörigen MOST-Netzwerks liegt damit etwa bei 4 W (vgl. Modellgleichung 3.6).

Ein Ethernet-Stern mit fünf Knoten (einer davon ein Switch) vernetzt diese Komponenten unter einer Leistungsaufnahme von ca. 2 W (vgl. Tabelle 3.1). Hier ist bereits Ethernet ohne EEE energieeffizienter als MOST und halbiert den Leistungsbedarf. Da hier hauptsächlich Audioströme von Quellen (Digitalradio, Mobiltelefone, CDs) an Senken (Verstärker) übertragen werden, ist mit asymmetrischer Auslastung zu rechnen. Die Datenrate ist im Audiobereich gering (üblicherweise komprimiert 256 kbit/s bis 320 kbit/s). In diesem Beispielnetzwerk läge die Auslastung unter einem Prozent, so dass auf allen Links über 80 % Leistungsreduktion möglich wäre. Damit läge der Leistungsbedarf des Netzwerks nur noch bei 0,5 W. Kämen höhere Datenraten vor (Navigationssystem, Displays, HD-Videos), läge das Sparpotential aufgrund der asymmetrischen Natur immer noch bei 45 %, und der Leistungsbedarf des Netzwerks damit bei knapp 1 W.

Streaming von vier Kameras

Das Streaming der Bilder mehrerer Kameras an ein zentrales Steuergerät war das Pionierprojekt der Einführung von Ethernet im Fahrzeug [54]. Die Bilder von vier Kameras am Fahrzeug, jeweils vorne, hinten und an den Außenspiegeln, lassen sich zu einem Draufsicht-Bild des Fahrzeugs zusammenrechnen. Dieses Bild hilft dem Fahrer die Umgebung seines Fahrzeugs, etwa beim Einparken, abzuschätzen. Früher mittels LVDS (*Low Voltage Differential Signaling*) angebunden, werden diese Kameras heute über Ethernet mit dem Steuergerät vernetzt, da Ethernet eine maßgebliche Kostenersparnis gegenüber LVDS darstellt. Die Topologie besteht aus vier Kameras und dem Steuergerät, das gleichzeitig den Switch darstellt. Aus Netzwerksicht ist das Beispiel sehr ähnlich dem vorherigen. Auch hier beträgt die Leistungsaufnahme des Netzwerks ohne Verwendung von EEE 2 W. Auf den vier Links wird jeweils mit bis zu 40 Mbit/s von der Kamera zum Steuergerät gestreamt. Der Rückkanal hat

vernachlässigbare Datenraten. Damit ergibt sich auch bei diesem System fast eine Halbierung der notwendigen Leistungsaufnahme auf ca. 1,1 W durch Energy Efficient Ethernet.

Backbone-Netz

In Backbone-Netzen zukünftiger Fahrzeugarchitekturen ist mit einer gewissen Wahrscheinlichkeit keines der Kriterien erfüllt. Wäre die Auslastung gering, bräuchte man kein Ethernet. Die Paketgröße ist im Backbone nicht vorhersagbar, es wird große und kleine Pakete geben. Die Linkauslastung wird sehr wahrscheinlich symmetrisch verteilt sein, da sich hier Datenströme verschiedener Quellen und Senken überlagern. Das legt nahe, dass EEE auf Backbone-Links wenig bis keine Einsparungen erlaubt.

3.5.3 Fazit

Tabelle 3.2 zeigt den in den vorherigen Kapiteln ermittelten Leistungsbedarf klassischer Bustechnologien und 100BASE-T Ethernet im Vergleich zueinander. Ethernet erfordert pro Knoten mehr als die zehnfache Leistung eines CAN-Busses, bietet aber auch das zweihundertfache seiner Datenrate und Kollisionsfreiheit, und ermöglicht damit eine Vielzahl neuer Anwendungsfälle. Verglichen mit MOST und FlexRay ist der Unterschied in der Leistungsaufnahme nicht mehr so bedeutend. Unter den im vorherigen Abschnitt gezeigten Bedingungen kann an dieser Stelle der Leistungsbedarf von Ethernet pro Knoten mittels EEE sogar niedriger ausfallen als bei FlexRay, bei gleichzeitig zehnmal höherer Datenrate. Rein aus Sicht der Energieeffizienz kann EEE also beim Einsatz von Ethernet im Automobil große Vorteile bringen. Es müssen jedoch zuvor weitere Aspekte untersucht werden, die im Rahmen dieser Arbeit nicht

Tabelle 3.2: Typische Leistungsaufnahme pro Knoten im Vergleich

Netzwerk	Leistung/Knoten	Datenrate
CAN	40 mW/Knoten	500 kbit/s geteilt
FlexRay	200 mW/Knoten	10 kbit/s geteilt
MOST25	765 mW/Knoten	25 Mbit/s geteilt
100BASE-TX	500 mW/Knoten	100 Mbit/s dediziert

betrachtet wurden. Zum einen muss die technische Umsetzbarkeit von EEE bei den für die Automobilindustrie entwickelten PHY-Technologien 100BASE-T1 (*BroadR-Reach*) und dem in der Standardisierung befindlichen 1000BASE-T1 betrachtet werden. Zum anderen muss der Einfluss der LPI-Übergangszeiten auf Dienstgütemechanismen wie Bandbreitenreservierung, Zeitsynchronisation und Traffic Shaping nach IEEE 802.1Q sowie auf die aktuellen Arbeiten im Bereich Ethernet TSN (*Time Sensitive Networking*) untersucht werden. Auch unterschiedliche Ausprägungen des Datenverkehrs im Bordnetzwerk sind noch zu untersuchen. Kunze *et al.* haben in [50] die im Rahmen dieser Arbeit entwickelten Modelle als Ausgangspunkt für die Modellierung von EEE bei *BroadR-Reach* benutzt und die hier vorgenommene Betrachtung von periodischem Datenverkehr noch um andere Modelle erweitert. Dabei ziehen sie auch den Schluss, dass beispielsweise bei blockweisem oder gebündeltem Datenverkehr die Effizienz von EEE ansteigt. Werden die *Idle*-Zeiten jedoch länger, ist irgendwann das Abschalten des Steuergeräts effektiver und damit der Einsatz von Teilnetzbetrieb.

4 Abschalten vernetzter Steuergeräte

4.1 Teilnetzbetrieb – Entstehung und verwandte Arbeiten

In Kapitel 2 wurde das selektive Abschalten von Steuergeräten als Maßnahme zur Senkung des Energiebedarfs im Bordnetz angesprochen. Dies wird als Teilnetzbetrieb bezeichnet. Der Name liegt wohl begründet in der Tatsache, dass bei heutigen Fahrzeugen während des Betriebs stets alle Steuergeräte (also das gesamte Netzwerk) eingeschaltet sind und gemeinsam, koordiniert durch das Netzwerkmanagement, heruntergefahren werden, sobald das Fahrzeug abgestellt wird. Der Gedanke, nicht benötigte Steuergeräte abzuschalten, also nur *Teile* des Steuergerätenetzwerks zu betreiben, entstand erst in jüngerer Zeit mit der immer weiter steigenden Anzahl elektronischer Komponenten und dem wachsenden Bedarf, Autos energieeffizienter zu gestalten. 2011 äußerten sich die Leiter der Entwicklung der elektrischen/elektronischen Architektur der fünf größten, deutschen Automobilhersteller im Rahmen des 15. Fachkongresses „Fortschritte der Automobilelektronik“ in Ludwigsburg das gemeinsame Interesse, Teilnetzbetrieb einzuführen und zu standardisieren [1]. Ein Gremium, welches sich aus Vertretern der Automobilhersteller und Halbleiterzulieferer zusammensetzte, begann 2011 mit der Ausarbeitung eines Konzeptes zur Realisierung von CAN-Teilnetzbetrieb [27], welches später in ISO 11989-6 überführt wurde [82]. Hierauf wird in Abschnitt 4.5 eingegangen. Das AUTOSAR-Konsortium rief die *Efficient Energy Management* (EEM) Arbeitsgruppe ins Leben [84] und konnte so auch die softwareseitigen Voraussetzungen treffen, CAN-Teilnetzbetrieb zu unterstützen (vgl. Abschnitt 4.4). Der Automobilhersteller Audi führt Teilnetzbetrieb in der nächsten Fahrzeuggeneration serienmäßig ein [27]. Zukünftige Netzwerktechnologien wie Ethernet standen bei der Standardisierung nicht im Fokus der Aktivitäten rund um Teilnetzbetrieb. Ethernet war zu neu, zu punktuell und die Industrie hatte mit grundlegenderen Fragen um dessen Einführung genügend zu tun. FlexRay war aufgrund seines zeitsynchronen

Charakters ebenfalls nicht in den Konzepten um Teilnetzbetrieb enthalten. Während CAN-Teilnetzbetrieb also weitgehend durch die ISO und das AUTOSAR-Konsortium abgedeckt wurde, entstand Raum für Forschungsarbeiten rund um die Befähigung anderer Netzwerktechnologien zum Teilnetzbetrieb. Christoph Schmutzler behandelt in seiner Dissertation Teilnetzbetrieb bei FlexRay ausführlich und stellt einen Lösungsansatz dafür vor [72].

Ethernet-Teilnetzbetrieb wird im Rahmen der vorliegenden Arbeit ausführlich beleuchtet. Die beschriebenen Ansätze wurden teilweise erstmals 2011 im Rahmen des durch das Bundesministerium für Bildung und Forschung geförderten Forschungsvorhabens SEIS (Sicherheit in eingebetteten, IP-basierten Systemen) veröffentlicht [97, 91], patentiert und auf Konferenzen und Fachtagungen vorgestellt. Verwandte Arbeiten im Bereich Ethernet-Teilnetzbetrieb haben vornehmlich die Weckfähigkeit von Ethernet-Steuergeräten zum Gegenstand. Seyler *et al.* beschreibt ein Verfahren, um Ethernet-Steuergeräte mittels Frequenzmultiplex zu wecken [74]. Sürmann und Müller von der Halbleiterfirma NXP Semiconductors beschrieben 2014 ein Weckverfahren für Ethernet auf Basis der Erkennung von Leitungssignalen [78]. Der enge Kontakt innerhalb der (damals kleinen) Ethernet-Gemeinschaft innerhalb der Automobilindustrie war hilfreich, um Konzepte firmenübergreifend zu diskutieren. So haben die im Rahmen dieser Arbeit entstandenen, und mit den Halbleiterherstellern diskutierten Ansätze ihren Teil zur Findung eines in Zukunft industrieweiten Weckkonzeptes für Ethernet-Komponenten beigetragen. Die Weckfähigkeit von Steuergeräten über Ethernet wird später im Detail beleuchtet (Kapitel 5). In diesem Kapitel werden im weiteren Verlauf die Grundzüge von Teilnetzbetrieb und seine Auswirkungen auf Hardware und Software von Steuergeräten beschrieben. In Kapitel 5 wird ausgearbeitet, welche Implikationen Ethernet als Vernetzungstechnologie auf Teilnetzbetrieb hat und welche Unterschiede sich zu CAN-Teilnetzbetrieb ergeben. In Kapitel 6 wird das *Energy Detect Modul* vorgestellt, welches eine selektive Weckfähigkeit von Ethernet-Steuergeräten über das Netzwerk ermöglicht. In den darauffolgenden Kapiteln werden zwei Netzwerkmanagement-Konzepte zur software- und protokollseitigen Unterstützung von Ethernet-Teilnetzbetrieb beschrieben.

4.2 Voraussetzungen für Teilnetzbetrieb

In einer vernetzten Architektur, in der Steuergeräte über Protokolle miteinander kommunizieren und Abhängigkeiten zueinander besitzen, ist das Abschalten einzelner

Komponenten nicht ohne weiteres möglich. In heutigen Fahrzeugnetzen sind alle Steuergeräte stets gemeinsam wach und können miteinander kommunizieren. Damit stellt sich die Frage nach der Koordination des Zustands einzelner Knoten im Netzwerk erst gar nicht, einzig ein gemeinsames Aufwachen, Wachbleiben und wieder Abschalten ist notwendig. Dies wird über das Netzwerkmanagement des jeweiligen Netzwerksystems gesteuert, wie in Abschnitt 4.3 beschrieben wird. Selektiv einzelne Steuergeräte oder Steuergerätegruppen während des Betriebs im Netzwerk abzumelden und auszuschalten erfordert geeignete Maßnahmen zur Koordination. Grundsätzlich gilt dabei, dass sich Steuergeräte (oder Teilnetze), die zu einem Zeitpunkt nicht benötigt werden, abschalten können. Sobald eine durch sie realisierte Funktion oder bereitgestellte Information wieder erforderlich ist, müssen sie aufgeweckt werden. Dies hat einige Implikationen auf das Netzwerkmanagement zur Folge, welches jetzt nicht mehr nur den Zustand des gesamten Bussystems koordinieren muss, sondern den Zustand individueller Systeme.

Die folgende Auflistung zeigt einige Veränderungen und Voraussetzungen die für Teilnetzbetrieb grundlegend sind:

- Das Netzwerkmanagement muss Teilnetzbetrieb koordinieren und ermöglichen. Einzelne Steuergeräte bzw. Steuergerätegruppen müssen bei Bedarf durch das Netzwerkmanagement wachgehalten werden, nicht wie bisher der ganze Bus.

- Softwarekomponenten sind notwendig, die koordinieren und entscheiden zu welchem Zeitpunkt welches Steuergerät wachgehalten werden soll.

- Das Fehlermanagement muss Teilnetzbetrieb berücksichtigen. Bislang wird das Ausbleiben von Netzwerkbotschaften einer Komponente als Indiz für einen Ausfall oder eine Fehlfunktion gesehen. Bei Teilnetzbetrieb gilt dies nur noch in den Zeiträumen, in denen das jeweilige Steuergerät wach ist.

- Steuergeräte müssen selektiv aufweckbar sein. Das bei heutigen Bussen übliche Konzept, alle Steuergeräte gemeinsam über das Netzwerksignal zu wecken, erlaubt es individuellen Steuergeräten nicht, schlafen zu gehen. Deswegen sind Weckmechanismen notwendig, die es erlauben einzelne Steuergeräte gezielt aufzuwecken.

Die Aufgabe der Zustandskoordination obliegt dem Netzwerkmanagement. Zunächst wird daher auf den Begriff, die Aufgaben und die Funktionsweise des klassischen

Netzwerkmanagements bei heutigen Fahrzeugbussystemen, noch ohne die Realisierung von Teilnetzbetrieb, eingegangen. Daraufhin wird aufgezeigt, welche Veränderungen Teilnetzbetrieb für das Netzwerkmanagement mit sich bringt und wie diese durch die in der Automobilindustrie gebräuchlichen Standards umgesetzt werden. Anschließend wird die Realisierung von Teilnetzbetrieb für den CAN-Bus beschrieben.

4.3 Netzwerkmanagement bei Fahrzeug-Bussystemen

Bei heute existierenden Fahrzeugarchitekturen ist Teilnetzbetrieb noch nicht vorgesehen. Dennoch müssen vernetzte Steuergeräte ein- und ausgeschaltet werden, was eine zentrale Aufgabe des Netzwerkmanagements darstellt.

4.3.1 Aufgaben des Netzwerkmanagements

Netzwerkmanagement im Kontext der Automobilelektronik stellt im weitesten Sinne die Verfügbarkeit des Netzes und der Steuergeräte sicher. OSEK/VDX nennt im Teil OSEK-NM (ISO 17356-5) [46] die folgenden Aufgaben:

- Initialisierung der Netzwerkschnittstelle
- Start des Netzwerks
- Netzwerk-Konfiguration
- Verfahren zur Knotenüberwachung
- Erkennung, Verarbeitung und Signalisierung von Betriebszuständen des Netzwerks und der Knoten
- Lesen und Schreiben von netzwerk- und knotenspezifischen Parametern
- Unterstützung der Diagnosefunktionen
- Koordination globaler Betriebszustände (z. B. netzwerkweiter Ruhezustand)

Viele dieser Aspekte betreffen direkt das Ein- und Ausschalten von Steuergeräten, beispielsweise beim Start des Netzes oder beim Wechseln globaler Betriebszustände. Darauf wird im Folgenden eingegangen.

4.3.2 Betriebszustände des Netzwerks und der Steuergeräte

Steuergeräte in einem Fahrzeugbussystem *wachen* aufgrund verschiedener Ursachen *auf*, sobald das Fahrzeug, bzw. das jeweilige Subsystem im Fahrzeug aufwacht. Diese Ursachen können elektrische Signale, wie beispielsweise das Zündungssignal oder ein Signal der Fahrzeugentriegelung sein, oder aber Aktivität auf dem Busnetzwerk. Sobald ein Steuergerät aufwacht, weckt es auch den Bus, d.h. alle anderen Steuergeräte in seinem Netzwerk. Dabei wird ein Signal auf dem Busmedium gesendet, welches von allen anderen Steuergeräten als Weckursache erkannt wird. Während des Betriebs des Fahrzeugs führen nun alle Steuergeräte ihre jeweiligen Funktionen aus, d.h. sie empfangen Busnachrichten, werten analoge und digitale Eingänge aus, verarbeiten Daten, steuern analoge und digitale Ausgänge an (z. B. LEDs, Motoren, Pumpen, etc.) und senden ihrerseits Nachrichten mit Werten und Steuersignalen sowie Netzwerkmanagementbotschaften auf den Bus. Sobald das Fahrzeug abgestellt und die Zündung abgeschaltet wird, müssen innerhalb einer gewissen Zeit auch alle Steuergeräte *einschlafen*, d.h. sich herunterfahren und ausschalten, da ansonsten ihr gemeinsamer Strombedarf die Fahrzeugbatterie über wenige Stunden entladen würde. Im brancheneigenen Jargon wird dies als *Einschlafen des Fahrzeugs* bezeichnet.

Diese gemeinsame Koordination des Betriebszustandes wird über das Netzwerkmanagement auf dem Bussystem realisiert. Da alle Steuergeräte am Bus durch Busaktivität gemeinsam aufwachen, muss eine Busruhe ausgehandelt werden, an die sich alle halten, sodass der ganze Bus gemeinsam schlafen gehen kann, ohne dass Teilnehmer durch verbleibende Busaktivität erneut geweckt werden. Dieser gemeinsame Zustandswechsel muss abgestimmt werden, sodass alle Steuergeräte ihre Vorbereitungen zum Abschalten treffen können, und nach und nach das Senden von Busnachrichten einstellen, d.h. den Bus freigeben. Erst wenn alle Steuergeräte diesen Zustand erreicht haben und Busruhe herrscht, dürfen sie sich abschalten. Das ist notwendig, da ansonsten Steuergeräte, die sich abschalten während andere noch Botschaften auf dem Bus senden, sofort wieder aufgeweckt werden würden. Die Synchronisation erfolgt dabei über Busnachrichten, die speziell dem Netzwerkmanagement dienen.

Eine vereinfachte, prinzipielle Darstellung der Betriebszustände ist in Abbildung 4.1 dargestellt. Sobald die Anwendungssoftware eines Steuergerätes die Buskommunikation freigibt (z. B. weil das Zündungssignal ausgeschaltet wurde) wird in einen Zwischenzustand gewechselt, in dem das Steuergerät die letzten Nachrichten aus dem Puffer versendet, bis schließlich Busruhe einkehrt. Je nach Konzept wird dieser Zustand –

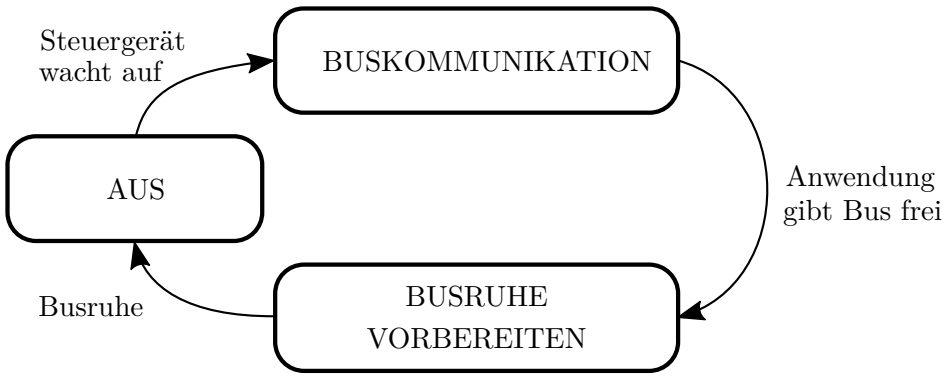


Abbildung 4.1: Vereinfachte Darstellung der Betriebszustände der Steuergeräte am Bus

d.h. die Absicht, den Bus herunterzufahren –, zwischen den einzelnen Knoten beispielsweise über dafür vorgesehene Bits in den Netzwerkmanagement-Botschaften (OSEK-Netzwerkmanagement) [46] synchronisiert oder durch Einstellen des periodischen Sendens von Netzwerkmanagement-Botschaften (AUTOSAR-Netzwerkmanagement) [5]. Sobald alle Knoten des Busses diesen Zustand erreicht haben und keine Busnachrichten mehr senden, können sich alle Steuergeräte abschalten, und der Bus schläft.

4.3.3 AUTOSAR-Netzwerkmanagement

Dieser Mechanismus funktioniert nur, wenn ihn alle Steuergeräte am Bus implementieren und alle sich gleich verhalten, unabhängig vom Steuergeräte- oder Softwarezulieferer. Sichergestellt werden kann das entweder durch entsprechende Vorgaben in den Lastenheften der einzelnen Komponenten, oder durch einen gemeinsam vereinbarten Softwarestandard. Das Netzwerkmanagement ist innerhalb OSEK/VDX und AUTOSAR jeweils fester Bestandteil des Kommunikationsstapels und dadurch industrieweit standardisiert. 2004 wurde die letzte Version 2.5.3 des Standards OSEK/VDX-NM (OSEK/VDX Network Management) in die ISO-Norm 17356-5 überführt [46]. Da die Arbeit des Gremiums hinter OSEK/VDX im AUTOSAR-Konsortium fortgesetzt wurde, und AUTOSAR heute den *de-facto* Standard in der Automobilsoftware darstellt, wird im Folgenden auf die Konzepte hinter dem AUTOSAR-Netzwerkmanagement eingegangen.

Der Steuergeräte- und Buszustand wird bei AUTOSAR durch ein Zusammenspiel verschiedener Module der Basissoftware verwaltet. Diese haben mit der Einführung

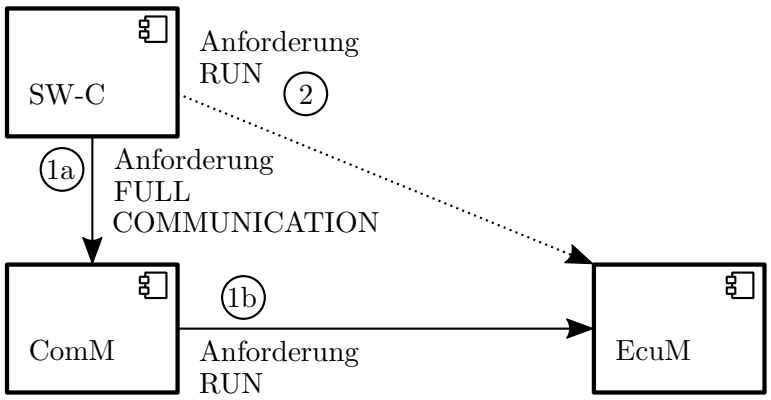


Abbildung 4.2: Anforderung des Betriebsmodus RUN beim *ECU State Manager* implizit durch den *Communication Manager* (1a, 1b) oder explizit durch eine Softwarekomponente (2)

von Teilnetzbetrieb in AUTOSAR Release 3.2 einige Veränderungen erfahren, auf die später noch eingegangen wird. Der folgende Überblick über das klassische Busmanagement mit AUTOSAR bezieht sich daher auf AUTOSAR Release 3.0 und früher. Zur Erleichterung der Wiedererkennbarkeit werden die im Standard verwendeten englischen Begriffe benutzt.

Der Zustand eines AUTOSAR-Steuergerätes wird durch den *ECU State Manager* (EcuM) abstrahiert. Er verwaltet und steuert die prinzipiellen Betriebszustände einer ECU, wie OFF, RUN und SLEEP, sowie die Übergänge dazwischen, also das hoch-, bzw. herunterfahren und initialisieren von Hardware, Betriebssystem und Basissoftware. [8] Der EcuM bietet eine Schnittstelle an, über die AUTOSAR-Softwarekomponenten (SW-C, die Applikationen in der AUTOSAR-Nomenklatur) den Betriebszustand RUN anfordern können. Wird dieser Zustand explizit oder implizit (durch einen anderen Teil der Basissoftware) angefordert, hält der EcuM das Steuergerät aktiv, ansonsten fährt er es herunter [8]. Im Zusammenhang mit Netzwerkkommunikation wird RUN implizit durch den *Communication Manager* (ComM) angefordert, sobald eine SW-C die Buskommunikation anfordert, wie in Abbildung 4.2 1a und 1b dargestellt. Der gestrichelte Pfeil (2) zeigt die explizite Anforderung des Zustandes RUN durch die SW-C.

Der ComM dient als Abstraktion einer darunter liegenden Kommunikationsschnittstelle [7]. Er verwaltet und steuert ihren Betriebs- und Kommunikationszustand und damit ihre Verfügbarkeit. Dadurch vereinfacht er das Ressourcenmanagement für SW-Cs. Diese fordern lediglich beim ComM einen Kommunikationsmodus für einen beliebigen Kommunikationskanal an. Dieser aktiviert oder deaktiviert den entsprechenden Kanal

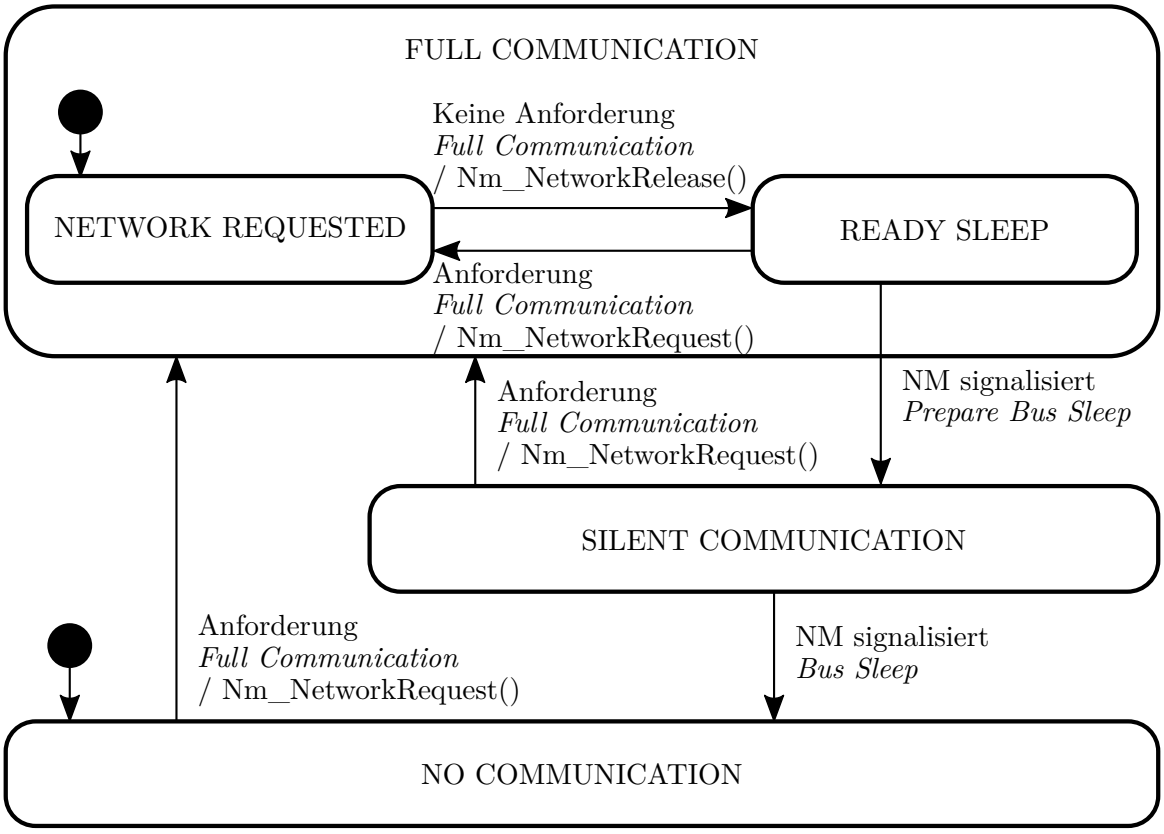


Abbildung 4.3: Vereinfachtes Zustandsdiagramm des *Communication Managers*

und verbirgt dabei jegliche hardware- und technologiespezifischen Aufrufe, die zur Initialisierung und Aufrechterhaltung der Kommunikationsfähigkeit der entsprechenden Netzwerkschnittstelle dienen. Insbesondere fällt darunter auch die Steuerung des (von der jeweiligen Bustechnologie abhängigen) Netzwerkmanagements auf dem Kanal. Damit ist der ComM ein zentrales Element bei der Koordination des Betriebszustandes des Busses (und des Steuergerätes selbst). Abbildung 4.3 zeigt eine vereinfachte Version des Zustandsdiagramms des ComM aus [7]. Der ComM hat für jede Netzwerkschnittstelle zwei wesentliche Zustände, NO COMMUNICATION und FULL COMMUNICATION. Sobald bei ComM der Zustand FULL COMMUNICATION durch eine SW-C angefordert wurde, geschehen drei wesentliche Dinge: Zunächst fordert ComM beim EcuM den Betriebsmodus RUN an, wie im vorherigen Abschnitt beschrieben und in Abbildung 4.2 1b dargestellt, damit das Steuergerät sich nicht ausschaltet. Über ein weiteres Modul, den *State Manager* der jeweiligen Netzwerkschnittstelle (z. B. *CAN State Manager*) wird der Kommunikationskanal aufgebaut, d.h. die Netzwerkhardware aktiviert und initialisiert. Gleichzeitig wird beim AUTOSAR Netzwerkmanagement (NM) das gesamte Netzwerk angefordert. Damit wacht der ganze Bus auf, das NM läuft

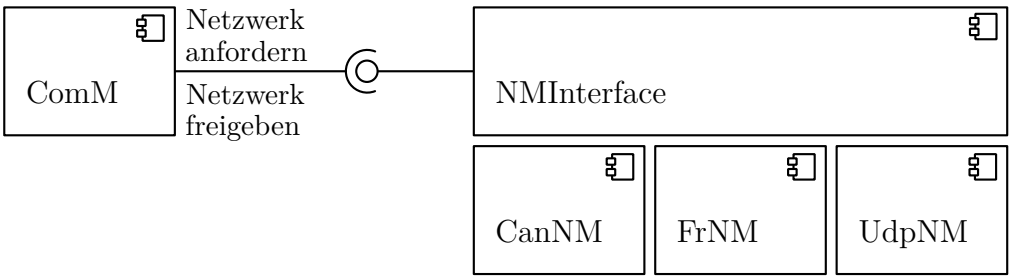


Abbildung 4.4: Das *NM-Interface* als Schnittstelle für den ComM zum Anfordern und Freigeben des Netzwerks; die spezifischen NM-Komponenten setzen dies auf dem jeweiligen Bussystem um [10]

an und koordiniert den gemeinsamen Buszustand (d.h. sorgt über einen geeigneten Mechanismus dafür das alle Steuergeräte am Bus wach bleiben und sich nicht abschalten). Bei CAN erfolgt dies beispielsweise über zyklisch versendete spezielle CAN-Botschaften (Netzwerkmanagement- oder NM-Botschaften).

Fordert keine SW-C mehr FULL COMMUNICATION beim ComM an, beginnt die Koordination des Einschlafens. Wie in Abbildung 4.3 dargestellt, verbleibt ComM noch im Zustand FULL COMMUNICATION, gibt aber beim NM das Netzwerk frei. Das NM des Steuergerätes hält jetzt seinerseits den Bus nicht länger wach (bei CAN wird beispielsweise das Senden von NM-Botschaften eingestellt). Das bedeutet, das Steuergerät ist nun bereit, einzuschlafen, der Bus wird aber noch durch andere Steuergeräte wachgehalten. Erst wenn alle Steuergeräte den Bus freigegeben haben, signalisiert das NM Busruhe beim ComM (*Prepare Bus Sleep* in Abbildung 4.3). Dies geschieht auf allen Steuergeräten annähernd gleichzeitig. Der ComM wechselt jetzt in einen Zwischenzustand SILENT COMMUNICATION und deaktiviert beim *State Manager* des Netzwerks die Sendefunktionen. Nachdem das NM schließlich *Bus sleep* signalisiert hat, wechselt ComM in den Zustand NO COMMUNICATION, und hört seinerseits auf, beim EcuM den Zustand RUN anzufordern. Dieser schaltet dann das Steuergerät aus. Im Zusammenspiel mit dem Netzwerkmanagement des jeweiligen Bussystems wird somit der Betriebszustand aller Steuergeräte am Bus koordiniert.

Der *Network Manager* bei AUTOSAR stellt die Kernfunktionalität des Netzwerkmanagements dar. Wie im vorherigen Abschnitt beschrieben, nutzt der ComM das Netzwerkmanagement, um Betriebszustand und Einschlafphase zu koordinieren. Während er nur eine Abstraktion des Betriebszustandes bietet, stellt das NM die Realisierung der Zustandskoordination dar. Der ComM gibt das Netzwerk beim NM frei und wartet so lange bis dieser ihm mitteilt, dass nun niemand mehr den Bus wachhalten möchte [10].

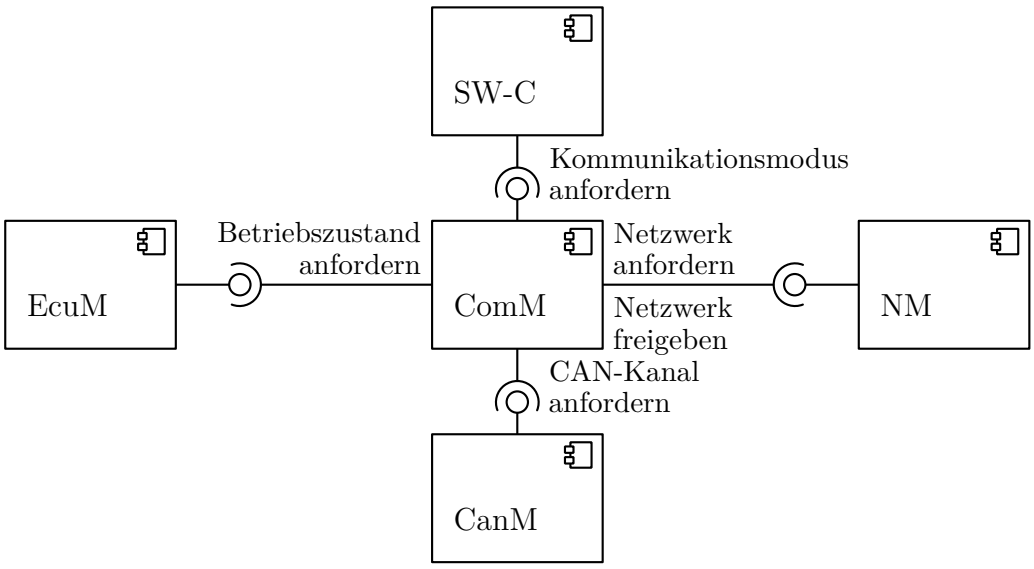


Abbildung 4.5: Zusammenspiel der an der Koordination des Buszustandes beteiligten AUTOSAR-Komponenten (Beispiel CAN)

Diese Koordination auf dem Bus erfolgt durch geeignete Mechanismen, spezifisch für jedes Bussystem. Dementsprechend gibt es für jede Bustechnologie im AUTOSAR-Standard einen eigenen *Network Manager*, jeweils untergeordnet unter dem generischen *NM Interface*, welches als gemeinsame Schnittstelle nach oben dient (vgl. Abbildung 4.4). Es gibt NM-Spezifikationen für FlexRay [9], CAN [5] und seit Release 4.0 für Ethernet [16]. Beispielhaft wird im Folgenden das *CAN Network Management* (CanNM) beschrieben.

Die Grundlage des CanNM ist die periodische NM-Botschaft (NM-PDU[1]). Dabei handelt es sich um einen leeren CAN-Header, der periodisch von allen NM-Instanzen am Bus gesendet wird. Knoten, die diese NM-PDUs senden, zeigen damit an, dass sie den Bus wachhalten möchten. Sobald der ComM das Busnetzwerk (wie im vorherigen Abschnitt beschrieben) freigibt, hört CanNM mit dem periodischen Senden der NM-PDUs auf und wartet darauf, dass auch von den anderen Steuergeräten am Bus keine mehr gesendet werden, in anderen Worten, bis jeder den Bus freigegeben hat. Dies wird durch einen Timer realisiert, der mit jeder empfangenen NM-PDU zurückgesetzt wird. Läuft er ab, bedeutet das, dass niemand mehr NM-PDUs auf dem Bus sendet. Dann wechselt CanNM in den Zustand BUS SLEEP und signalisiert dies beim ComM. Das für Ethernet spezifizierte UdpNM (seit Release 4.0) [16] basiert auf dem identischen Prinzip, NM-PDUs werden dort als UDP Broadcastpakete versendet.

[1] *PDU: Protocol Data Unit*

Die Koordination des busweiten Betriebszustandes der Steuergeräte bei AUTOSAR ist ein komplexes Zusammenspiel verschiedener durch AUTOSAR spezifizierter Komponenten (vgl. Abb. 4.5). Die in diesem Kapitel dargestellte Kurzbeschreibung dieser Komponenten und deren Zusammenspiel soll nun noch einmal in wenigen Sätzen zusammengefasst werden.

- SW-Cs (Softwarekomponenten), die über den Bus kommunizieren wollen, fordern beim ComM (*Communication Manager*) FULL COMMUNICATION an.
- Der ComM fordert beim EcuM (*ECU State Manager*) den Betriebsmodus RUN an, so dass das Steuergerät wach bleibt.
- Der ComM startet beim *State Manager* der Netzwerkschnittstelle (z. B. CanM) das Businterface und initialisiert die Hard- und Softwarekomponenten, so dass Netzwerkkommunikation möglich ist.
- Der ComM fordert beim zugehörigen *Network Management* (z. B. CanNM) das Netzwerk an.
- Das NM hält den ganzen Bus aktiv, z. B. durch periodisches Senden von NM-PDUs (CanNM). Dadurch wachen alle Steuergeräte am Bus auf und beginnen ihrerseits mit dem Senden von NM-PDUs. Damit ist der normale Betriebszustand des Busses hergestellt.
- Fordern die SW-Cs nicht mehr FULL COMMUNICATION beim ComM an, beginnt das koordinierte Einschlafen des Busses.
- Der ComM gibt das Netzwerk bei NM wieder frei, und wartet darauf, dass NM Busruhe signalisiert.
- NM stellt das Wachhalten des Busses (z. B. das periodische Senden von NM-PDUs bei CanNM) ein. Nach und nach findet dies bei allen Steuergeräten statt, bis keines mehr NM-PDUs auf dem Bus sendet.
- Sobald dies der Fall ist, signalisiert NM Busruhe beim ComM. Dies passiert bei allen Steuergeräten im Netzwerk ungefähr gleichzeitig.
- ComM deaktiviert daraufhin das Netwerkinterface, und hört auf, beim EcuM den Betriebsmodus RUN anzufordern.
- Der EcuM schaltet das Steuergerät ab.

Durch diesen Mechanismus ist sichergestellt, dass sich die Steuergeräte eines Busses erst abschalten, sobald der Zustand der Busruhe erreicht ist, also, sobald alle Steuergeräte bereit sind, einzuschlafen. Die hier dargestellten und durch den AUTOSAR-Standard vereinheitlichten Mechanismen beschreiben das Zustandsmanagement der heute im Fahrzeug verwendeten Bussysteme. Im Vordergrund steht dabei, beim Aufwachen des Fahrzeugs alle Steuergeräte über den Bus zu wecken, wach zu halten, und beim Einschlafen des Fahrzeugs koordiniert herunterzufahren. Teilnetzbetrieb wird hier noch nicht berücksichtigt und stellt eine Erweiterung dieser Mechanismen dar. Es wird wird schnell deutlich, dass Teilnetzbetrieb – also das Ein- und Ausschalten von Steuergeräten zur Laufzeit – einen massiven Eingriff in die Koordination des Buszustandes darstellt. Teilnetzbetrieb muss also als Teil des Netzwerkmanagements vorgesehen werden, bzw. das bestehende Netzwerkmanagement erweitern. Der Buszustand muss nun nicht mehr global koordiniert werden, sondern für einzelne Steuergeräte oder Gruppen von Steuergeräten (Teilnetze) individuell. Das AUTOSAR-Konsortium hat mit Release 3.2 begonnen, Konzepte für Teilnetzbetrieb in den Standard einfließen zu lassen. Das bislang einzige Bussystem, das auch technologisch Teilnetzbetrieb – also das selektive Wecken von Steuergeräten – ermöglicht, ist CAN.

4.4 Teilnetzbetrieb bei AUTOSAR

Die Unterstützung von Teilnetzbetrieb im AUTOSAR Netzwerkmanagement wurde mit AUTOSAR Release 3.2 und 4.0 eingeführt. Wesentlich sind davon die in Abschnitt 4.3 vorgestellten Module ComM und CanNM betroffen. Teilnetze werden in AUTOSAR als *Partial Network Clusters* (PNC) bezeichnet [13]. PNCs sind dabei Bussystemübergreifend und Systemweit definiert. Der ComM wurde erweitert, so dass er nun für jedes PNC eine eigene Zustandsmaschine führt, zusätzlich zu der in 4.3.3 dargestellten für die einzelnen Kommunikationskanäle (physischen Netzwerke) des Steuergerätes. SW-Cs können nun beim ComM Teilnetze (also PNCs in der AUTOSAR-Terminologie) anfordern und freigeben. Für jeden angeforderten PNC setzt ComM ein Bit in einem Bitvektor, in dem jede Bitposition statisch jeweils einem PNC zugeordnet ist. Dieser Bitvektor wird vom Bus-Netzwerkmanagement (CanNM oder FrNM) in den *User*-Daten der periodisch versendeten NM-PDUs mitgesendet.

Das BusNM kombiniert die Zustandsbits der Teilnetze aus den NM-PDUs aller Knoten (externe Anforderungen) und des ComM (interne Anforderungen) und erhält so einen aggregierten Bitvektor, der die Information enthält, welche Teilnetze derzeit im System

angefordert werden. Dieser Bitvektor wird an den ComM signalisiert, der dementsprechend den Zustand aller Teilnetze verwaltet. Für jede ECU wird statisch konfiguriert, welche PNCs für sie relevant sind. Anhand des mit den NM-PDUs mitgesendeten Bitvektors werden durch einen Filteralgorithmus alle NM-PDUs ausgefiltert (ignoriert) die für das jeweilige Steuergerät irrelevant sind.

Dabei gilt:

- NM-PDUs, bei denen ein Bit für einen PNC gesetzt ist, welcher für die ECU relevant ist, werden berücksichtigt.

- NM-PDUs, bei denen kein Bit für einen PNC gesetzt ist, welcher für die ECU relevant ist, werden ignoriert.

Infolgedessen erhält das Netzwerkmanagement nur NM-PDUs für angeforderte und für das jeweilige Steuergerät relevante Teilnetze. An dieser Stelle wird deutlich, warum der Rest des Netzwerkmanagements trotz Teilnetzbetrieb unverändert bleiben kann. Die ausgefilterten NM-PDUs werden vom NM nicht berücksichtigt und der NM-Timeout wird dadurch nicht zurückgesetzt. Sobald also keines der für ein Steuergerät relevanten PNCs mehr angefordert wird, werden keine NM-PDUs mehr empfangen, und das Steuergerät wird über den bereits in Abschnitt 4.3.3 beschriebenen Mechanismus ganz normal abgeschaltet. Die Synchronisation des Buszustandes erfolgt nun nicht mehr busweit, sondern innerhalb der PNCs.

4.5 Weckfähiger CAN-Bus

Mit dem AUTOSAR-Netzwerkmanagement sind softwareseitig die Voraussetzungen für CAN-Teilnetzbetrieb erfüllt. Aber auch die Bustechnologie selbst musste angepasst werden. Vor der Einführung von Teilnetzbetrieb in der Fahrzeugvernetzung erlaubten die auf dem Markt verfügbaren CAN-Transceiver lediglich das Wecken des Steuergeräts bei der Erkennung von Aktivität auf der Busleitung. Dieser Mechanismus erlaubt das in 4.3.3 beschriebene, busweite Wecken von Steuergeräten, da alle Knoten mit der Busleitung verbunden sind, und Busaktivität von allen detektiert wird. Das Prinzip soll im Folgenden kurz beschrieben werden.

4.5.1 Busweites Wecken durch Busaktivität

Ein CAN-Transceiver ist ein Hardware-Baustein, der die ihm über eine digitale Schnittstelle übergebenen CAN-Botschaften (z. B. vom Microcontroller) auf dem Bus versendet, bzw. vom Bus empfangene CAN-Botschaften an den Microcontroller weitergibt. Einige CAN-Transceiver bieten zusätzlich weitere Funktionen an, beispielsweise die Erkennung und Signalisierung von Busaktivität (z. B. NXP TJA1041 [64]). Diese Transceiver verfügen über Schaltungsteile, die bei einem dominanten Buszustand einen elektrischen Ausgangspin auf die Batteriespannung durchschalten, um beispielsweise die Spannungsversorgung des Steuergerätes zu aktivieren.

Die folgende Beschreibung bezieht sich exemplarisch auf den CAN-Transceiver TJA1041 des Herstellers NXP Semiconductors [64]. Während das Steuergerät schläft, sind die Spannungsregler ausgeschaltet. Der Schaltungsteil des Transceivers, der für die Aktivitätserkennung zuständig ist, wird direkt durch die Batteriespannung versorgt, und benötigt einen geringen Ruhestrom von etwa 30 µA. Sobald ein dominanter Buszustand für eine Zeitspanne von ca. 5 µs erkannt wird, wechselt der Transceiver in den normalen Betriebszustand und aktiviert über den *Inhibit-Pin* (INH) die Spannungsregler. Microcontroller und Peripherie starten infolgedessen, und das Betriebssystem (AUTOSAR) beginnt mit der Boot-Routine. Abbildung 4.6 zeigt in einem vereinfachten Blockdiagramm wie der CAN-Transceiver in die Hardware-Architektur eingebettet ist. Dieser Mechanismus erlaubt jedoch kein selektives Wecken, was aber Voraussetzung für einen Teilnetzbetrieb ist.

4.5.2 Selektives Wecken bei CAN-Teilnetzbetrieb

ISO 11898-6 definiert den 2013 neu spezifizierten Mechanismus, über den ein CAN-Knoten selektiv geweckt werden kann [47]. Dabei wird ein *Wake-Up Frame* (WUF) verwendet, der vom Transceiver eines schlafenden CAN-Knotens ausgewertet wird und gegebenenfalls zum Aufwecken des Knotens führt. Im Datenfeld des WUF befindet sich eine Bitmaske. Auch hier steht jedes Bit für eine Gruppe an CAN-Knoten, die gemeinsam aufwachen sollen. Ein kompatibler CAN-Transceiver vergleicht das Datenfeld eines empfangenen WUF mit einer internen, statisch konfigurierten Bitmaske. Enthält der empfangene Bitvektor mindestens an einer der in der Maske konfigurierten Bitpositionen eine Eins, wechselt der Transceiver aus dem Schlafzustand in den normalen Betriebszustand und weckt das Steuergerät über den *Inhibit*-Pin auf, wie im

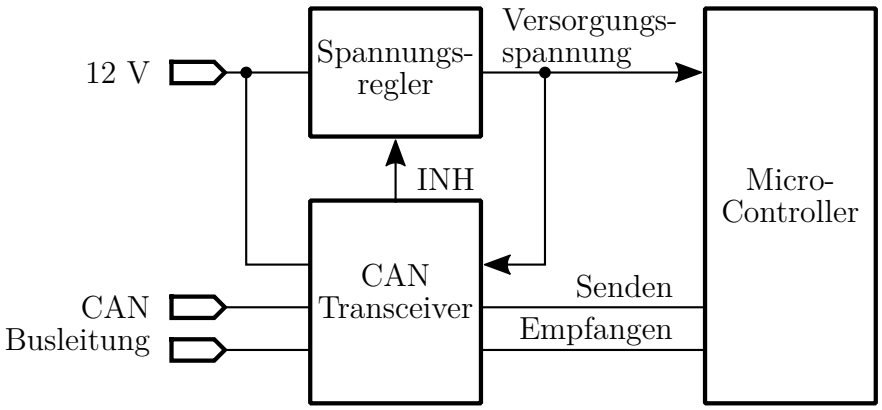


Abbildung 4.6: Der CAN-Transceiver aktiviert bei Busaktivität mit dem *Inhibit*-Signal (INH) den Spannungsregler [64]

vorherigen Abschnitt beschrieben. Dabei wird jeder Gruppe eine feste Bitposition in der Maske zugeordnet und bei den Knoten, die Teil dieser Gruppe sind, an dieser Stelle in der Maske des Transceivers eine Eins gesetzt. Durch senden einen WUF, in dessen Datenfeld das entsprechende Bit ebenfalls gesetzt ist, wachen die entsprechend konfigurierten Knoten auf, während der Rest weiterschläft. Das Empfangen und Auswerten von *Wake-up-Frames* erfordert einen höheren Ruhestrom als das vorher beschriebene Erkennen von Busaktivität. Das Datenblatt des CAN-Transceivers TJA1145 des Herstellers NXP Semiconductors gibt an dieser Stelle einen Batteriestrom von etwa 400 µA an [63].

Das Konzept ist kompatibel mit den *Partial Network Clusters* des AUTOSAR-Standards (vgl. Abschnitt 4.4). Der Bitvektor, der in den NM-Botschaften mitgesendet wird, lässt sich direkt auf den Bitvektor im *Wake-up-Frame* nach ISO 11898-6 abbilden. Die periodischen NM-Botschaften werden dadurch zu *Wake-up-Frames* auf dem CAN-Bus. Fordert das Netzwerkmanagement einen *Partial Network Cluster* an, ist das entsprechende Bit im Vektor der gesendeten NM-Botschaften gesetzt, und die CAN-Steuergeräte, bei denen das jeweilige Bit in der Maske des Transceivers gesetzt ist, wachen auf.

Zusammen mit der Unterstützung durch AUTOSAR sind für den CAN-Bus sämtliche Voraussetzungen für Teilnetzbetrieb erfüllt. Diese lassen sich jedoch nicht ohne Weiteres auf Ethernet übertragen. Im Folgenden wird auf die grundlegenden Unterschiede und deren Auswirkungen eingegangen.

5 Teilnetzbetrieb bei Ethernet

5.1 Voraussetzungen

Teilnetzbetrieb bei Ethernet erfordert andere Ansätze als Teilnetzbetrieb auf dem CAN-Bus, bedingt durch technologische und architekturelle Unterschiede beider Netzwerktechnologien. Allgemein gelten die gleichen, allgemeinen Voraussetzungen, die in Abschnitt 4.2 bereits aufgezeigt wurden – eine Möglichkeit, den gewünschten Zustand einzelner Steuergeräte oder Steuergerätegruppen über das Netzwerk zu koordinieren, und Steuergeräte selektiv Wecken zu können und wach zu halten. Die Entscheidung, welches Gerät zu welchem Zeitpunkt geweckt wird, sowie die Berücksichtigung durch Diagnose und Fehlermanagement sind von der Netzwerktechnologie unabhängig. Ansätze und Verfahren, die bei Ethernet-Teilnetzbetrieb zur Koordination des Steuergeräte-Zustandes dienen, sollten dabei zu dem durch AUTOSAR standardisierten Netzwerkmanagement kompatibel sein, bzw. darin integrierbar sein. Eine Abstraktion des Dienstes Teilnetzbetrieb von der darunterliegenden Netzwerktechnologie ist wünschenswert, damit höher liegende Schichten Teilnetzbetrieb unabhängig davon koordinieren können, in welchen Netzwerken die jeweiligen Steuergeräte partitioniert sind. Das selektive Wecken von Ethernet-Steuergeräten ist eine weitere Voraussetzung, die im Kontext automobiler Netzwerke betrachtet und erfüllt werden muss.

5.2 Unterschiede zu CAN-Teilnetzbetrieb

CAN und Full-Duplex-Ethernet sind zwei grundlegend verschiedene Netzwerktechnologien mit unterschiedlicher Topologie. Deswegen sind auch die jeweiligen Ansätze für selektives Wecken und die Zustandskoordination im Netzwerk nicht ohne Weiteres übertragbar. CAN ist ein Bus, d.h. alle Knoten sind über ein gemeinsames

Medium miteinander verbunden. Ethernet hingegen ist ein (schleifenfreies) Punkt-zu-Punkt-Netzwerk. Hier sind Knoten nur mit jeweils einem direkten Nachbarn über ein gemeinsames Medium (Kabel) miteinander verbunden. Bei CAN erreichen also Frames auch die Transceiver abgeschalteter (schlafender) Steuergeräte. Dies gilt auch für Netzwerkmanagement-Pakete und die in Abschnitt 4.5 angesprochenen Wake-Up-Frames. Bei Ethernet werden Pakete hingegen nur über aktive Links versendet (auch bei Broadcast-Nachrichten), d.h. an Steuergeräte, bei denen mindestens die Netzwerkschnittstelle aktiv und der Link aufgebaut ist. Die Ethernet-Schnittstelle schlafender Steuergeräte aktiv zu halten, hätte einen zu hohen Ruhestrom (ca. 200 mW, vgl. 3.3) zur Folge, weswegen dies keine gute Lösung darstellt. Damit besteht keine Möglichkeit, schlafenden Ethernet-Knoten Pakete zukommen zu lassen, weswegen das beim CAN-Bus verwendete Konzept der Koordination und des Aufweckens mittels Wake-Up-Frames nicht anwendbar ist. Aus diesem Grund ist für Ethernet ein anderer Ansatz erforderlich. Im Rahmen dieser Arbeit wird ein nachbarbasiertes Netzwerkmanagementkonzept als Lösungsmöglichkeit beschrieben.

5.3 Selektives Wecken von Ethernet-Steuergeräten

Das Wecken von Steuergeräten über das Datennetzwerk ist eine der Grundvoraussetzungen des in den vorherigen Kapiteln beschriebenen Netzwerkmanagements. Auch ohne Teilnetzbetrieb ist es für das Aufwachen und Einschlafen des Fahrzeuges notwendig, um Steuergeräte auch unabhängig vom Zündungssignal wecken zu können. Für die Umsetzung von Teilnetzbetrieb ist es erforderlich, Steuergeräte selektiv Wecken zu können. Im vorherigen Kapitel wurde dargestellt, wie selektives Wecken bei CAN-Bussen funktioniert. Für Ethernet gibt es auch Möglichkeiten, Knoten selektiv zu wecken. Einige existierende Verfahren werden im Folgenden beschrieben und bewertet.

5.3.1 Weckleitung

Unter einer Weckleitung (*Wake-Up Line*) versteht man eine dedizierte, elektrische Leitung zwischen zwei Knoten, die in der Lage sein sollen, einander aufzuwecken. Über ein elektrisches Signal kann der weckende Knoten den schlafenden Knoten direkt einschalten, wenn beispielsweise die Weckleitung direkt als Wecksignal im Energiemanagement des Moduls verwendet wird. Hiermit sind viele Topologien denkbar, wie

beispielsweise Punkt-zu-Punkt oder das Wecken mehrerer schlafender Knoten durch einen einzelnen. Der Vorteil der Weckleitung ist das technisch sehr einfach umsetzbare Konzept, bei dem zudem kein Ruhestrom benötigt wird. Außerdem ist das Konzept vom Bussystem oder Netzwerk unabhängig und somit für alle Steuergeräte anwendbar. Der Nachteil der Weckleitung ist jedoch, dass ein zusätzliches elektrisches Signal für jedes Steuergerät notwendig ist, also weitere Kabel im Kabelbaum und zusätzliche Steckerkontakte. Der Kabelbaum ist jedoch eine der teuersten und aufwändigsten Komponenten im Automobil [17] Der Kabelbaum der Mercedes W220 S-Klasse besitzt etwa 1900 Leitungen mit einer Gesamtlänge von drei Kilometern, und 3800 Steckerkontakte. Die Automobilhersteller legen sehr viel Wert darauf, Gewicht und Kosten im Kabelbaum zu reduzieren. Aus diesem Grund ist das Wecken über Netzwerk der Weckleitung vorzuziehen.

5.3.2 Wake-on-LAN

Der Begriff *Wake-on-LAN* (WOL) bezeichnet allgemein das Wecken von LAN-Knoten über das Netzwerk und dies kann mit verschiedenen Methoden realisiert werden. Im üblichen Sprachgebrauch versteht man darunter jedoch den *de-facto* Netzwerkstandard zum Wecken von Ethernet-Hosts über ein sogenanntes *Magic Packet*. Diese Technologie wurde 1995 als Weißpapier durch den Halbleiterhersteller AMD (Advanced Micro Devices) publiziert [2].

Wake-on-LAN entstammt der Informationstechnik, genauer der Computervernetzung. Die Motivation dahinter war die Lösung eines Interessenkonflikts. IT-Abteilungen großer Unternehmen führen die Wartung und Sicherung von Arbeitsrechnern durch, während diese unbenutzt sind. Gleichzeitig gibt es das Bestreben, den durch die Geräte hervorgerufenen Stromverbrauch zu reduzieren, also Computer auszuschalten, wenn sie nicht benutzt werden. Initiativen wie *Energy Star* fördern die Senkung des Energiebedarfs durch IT-Geräte wie Privat- und Arbeitsplatzcomputer, auch durch die Forderung nach dem automatischen Herunterfahren oder Schlafengehen nach einer gewissen Zeit ohne Benutzereingaben.

Wake-on-LAN entstand also als Methode, um auch auf abgeschaltete oder schlafende Computer außerhalb der regulären Benutzungszeiten ferngesteuert zugreifen zu können, um Wartung und Sicherung durchführen zu können. In dem Weißpapier von AMD wird die Funktionsweise, das *Magic Packet* und die Integration in den Netzwerkchip sowie in die Computer- und Netzwerkinfrastruktur beschrieben.

Bei WOL-fähigen LAN-Knoten bleibt die Netzwerkschnittstelle (Controller und PHY, meist integriert in einem Netzwerkchip) und damit auch der Netzwerk-Link des Rechners aktiv, während Motherboard, sonstige Peripherie und Prozessor in einen Niedrigenergie- oder Ruhezustand wechseln. Somit können noch Pakete an die Netzwerkschnittstelle zugestellt werden. Um einen schlafenden Knoten zu wecken, wird ein sogenanntes *Magic Packet* an ihn versendet. Dabei handelt es sich um einen normalen Ethernet-Frame, der im Nutzdatenfeld eine sechsmalige Wiederholung der MAC-Adresse des schlafenden Knotens und eine Synchronisationsfolge enthält. An welcher Stelle im Nutzdatenfeld diese Sequenz steht, ist nicht definiert. Deswegen kann das Magic Packet auch als UDP-Datagramm über IP verschickt und somit auch über mehrere Netze hinweg gerouted werden. Der WOL-fähige Netzwerkcontroller empfängt und erkennt das *Magic Packet*, und aktiviert den Computer über das Energiemanagement des Motherboards. Bei PCs ist heute das Netzwerkinterface in der Regel über PCIe (Peripheral Component Interconnect Express) angebunden (z. B. Intel 82574 [44]). Der Netzwerkchip kann dann beim Erkennen eines an ihn adressierten *Magic Packet* ein *Power Management Event* auf dem PCIe-Bus auslösen, um das Motherboard aufzuwecken. Bei Geräten ohne PCIe-Bus (z. B. eingebetteten Systemen) erfolgt das Wecken in der Regel über einen elektrischen Ausgang des Netzwerkchips. Damit erlaubt WOL ein zielgerichtetes Aufwecken von LAN-Computern bei geringem Integrationsaufwand. Der Nachteil liegt jedoch in der Tatsache, dass die gesamte Netzwerkschnittstelle wach und der Netzwerklink bestehen bleiben muss. Ansonsten könnte das *Magic Packet* gar nicht erst zugestellt werden. Dies hat einen hohen Ruhestrom zur Folge, der in der Regel fast so hoch ist, wie bei normalem Betrieb der Netzwerkschnittstelle. Der Intel 82574 hat etwa im 100BASE-TX Modus eine Leistungsaufnahme von 296 mW ([44]). Im Wake-on-LAN-Modus sind es immer noch 171 mW. Etwas mehr Leistungsaufnahme erfordert der Microchip LAN 9420, mit 415 mW im Normalbetrieb (100BASE-TX) und 252 mW im WOL-Modus. Für einen Einsatz auf Steuergeräten im Automobil sind diese Werte um Größenordnungen zu hoch. Ein Aufrechterhalten des Netzwerks während das Auto schläft, würde die Batterie innerhalb weniger Stunden entladen.

5.3.3 Wake-on-Link

Viele Netzwerkadapter wie auch der Intel 82574 unterstützen als Weckereignis auch einen Wechsel des Link-Zustandes. Dabei signalisiert der Netzwerkcontroller ein Weckereignis an das Energiemanagement, sobald ein erfolgreicher Aufbau des Netzwerklinks erfolgt. Dabei gelten jedoch die gleichen Voraussetzungen hinsichtlich Ruhestrom

wie bei *Wake-on-LAN* und der Erkennung des *Magic Packet*. Der Netzwerkadapter muss für einen Linkaufbau aktiv bleiben, auch während der Host abgeschaltet ist.

5.3.4 Wecken über Erkennung von Link-Aktivität

Eine Möglichkeit, sich bei Ethernet den Vorteil des Weckens über das Netzwerk – anstatt über eine separate Weckleitung – zu erhalten, ohne wie bei WOL den Strombedarf einer aktiven Netzwerkschnittstelle in Kauf nehmen zu müssen, ist das Wecken durch Erkennen von Aktivität auf der Link-Leitung. Ähnlich wie bei CAN kann eine Zustandsänderung auf dem Medium als Weckereignis genutzt werden. Die Punkt-zu-Punkt-Topologie von Ethernet erlaubt dabei jedoch (im Gegensatz zum CAN-Bus) auch Selektivität, da nur jeweils der weckende und der schlafende Knoten physisch mit dem selben Medium verbunden sind. Im Folgenden werden einige unterschiedliche Weckverfahren mit Aktivitätserkennung vorgestellt und verglichen.

Dabei werden die hier aufgeführten Kriterien betrachtet:

- Integrationsaufwand: Was ist notwendig ist, um das jeweilige Verfahren auf einem Gerät mit Ethernet-Schnittstelle zu integrieren? Wie komplex ist das Verfahren?

- Validierbarkeit: Werden falsche Weckereignisse (etwa durch Störsignale) erkannt oder muss dies auf einer höheren Schicht abgefangen werden?

- Intrusivität: Ist ein Eingriff oder eine Modifikation der Netzwerktechnologie (z. B. des Leitungssignals) notwendig? Zieht dieser Eingriff Veränderungen oder Verletzungen des Standards oder Modifikationen der Netzwerkhardware nach sich?

- Ruhstrombedarf: Welche Komponenten müssen bei schlafenden Steuergeräten mit Strom versorgt werden, und welche Ruheleistung ist dadurch zu erwarten?

- Weckzeit: Wie groß ist die Zeitverzögerung zwischen dem Auftreten des Weckereignisses und dem Aufwachen des Gerätes?

Signalerkennung von Leitungscodes

Das Leitungssignal des benachbarten Netzwerkknotens selbst kann als Weckereignis verwendet werden. Dazu notwendig ist eine geeignete Schaltung auf dem zu weckenden Knoten, die das Signal auf der Netzwerkleitung erkennt und über das Energiemanagement des Moduls einen Wake-Up auslöst. Um einen Nachbarn aufzuwecken, muss ein Knoten die entsprechende Netzwerkschnittstelle aktivieren, so dass Leitungscodes auf dem Medium gesendet werden. Diese Form des selektiven Weckens wurde im Rahmen dieser Arbeit untersucht und realisiert, und wird in Kapitel 6 im Detail beschrieben.

Tabelle 5.1: Bewertung der Signalerkennung von Leitungscodes

Kriterium	Bewertung
Integrationsaufwand	Gering, Schaltungsteil zur Aktivitätserkennung kann diskret oder als Teil des PHY realisiert werden.
Validierbarkeit	Keine Validierung. Diese muss durch das Netzwerkmanagement vorgenommen werden.
Intrusivität	Keine Modifikation des Netzwerksignals oder der Leitungscodes. Diskrete Realisierung könnte Signalintegrität beeinflussen.
Ruhestrombedarf	Signalerkennung mit minimalem oder ohne Ruhestrombedarf möglich.
Weckzeit	Einzige Zeitkomponente ist die (elektrische) Erkennung des Signals, dadurch sehr kurze Weckzeit möglich (Mikrosekunden-Bereich).

Validierung von Leitungscodes

Die Auswertung von Leitungscodes hat gegenüber der reinen Erkennung den Vorteil, Störsignale wie Rauschen oder Crosstalk nicht als Weckereignis zu werten. Erst wenn eine Erkennung eines gültigen Signalverlaufs vorliegt, wird das Steuergerät aufgeweckt. Dies erfordert jedoch eine Erweiterung des Netzwerktransceivers um einen Modus, in dem das Leitungssignal passiv überwacht und die Validierung durchgeführt wird. Im einfachsten Falle kann wie bei der reinen Erkennung direkt die *Idle*-Codegruppe des Ethernet-Leitungscodes verwendet werden. Dies hat gegenüber der Definition eigener, spezieller Codegruppen den Vorteil, dass keine Veränderung des Leitungscodes notwendig ist, und somit die Kompatibilität zu IEEE-konformen Transceivern gewährleistet bleibt.

Tabelle 5.2: Bewertung der Validierung von Leitungscodes

Kriterium	Bewertung
Integrationsaufwand	Hoch, Validierung der Codes im Ethernet-Transceiver erfordert Modifikation des Hardwarebausteins.
Validierbarkeit	Validierung erfolgt, Wake-Up durch Störsignale ist somit unwahrscheinlich.
Intrusivität	Keine Modifikation des Netzwerksignals oder der Leitungscodes notwendig, außer wenn statt *Idle* spezielle Codegruppen verwendet werden.
Ruhestrombedarf	Permanente Versorgung der validierungsrelevanten Teile des Transceivers führt zu einem erhöhten Ruhestrombedarf.
Weckzeit	Mehrfache Wiederholung des Leitungscodes, Millisekunden-Bereich.

Signalerkennung und Validierung

Ein zweistufiges Wecken durch Signalerkennung und darauffolgende Signalvalidierung kombiniert die Vorteile beider Verfahren. Im ersten Schritt wird nur der Netzwerktransceiver durch die Erkennung von Aktivität auf der Leitung geweckt. Danach werden zunächst die Leitungscodes validiert, bevor das Steuergerät über das Energiemanagement geweckt wird. Dadurch gibt es, während das Steuergerät inaktiv ist, keinen erhöhten Ruhestrombedarf. Erst während der Validierungsphase muss der Netzwerktransceiver mit Strom versorgt werden. Störsignale führen zwar zu einem (kurzzeitigen) Aufwachen des Transceivers, werden aber als falsch erkannt und führen nicht zum Wecken des Steuergerätes.

Tabelle 5.3: Bewertung der Signalerkennung- und Validierung

Kriterium	Bewertung
Integrationsaufwand	Hoch, Validierung der Codes im Ethernet-Transceiver erfordert Modifikation des Hardwarebausteins.
Validierbarkeit	Validierung erfolgt, Wake-Up durch Störsignale ist somit unwahrscheinlich.
Intrusivität	Keine Modifikation des Netzwerksignals oder der Leitungscodes notwendig, außer wenn statt *Idle* spezielle Codegruppen verwendet werden.
Ruhestrombedarf	Signalerkennung mit geringem oder ohne Ruhestrombedarf möglich.
Weckzeit	Mehrfache Wiederholung des Leitungscodes, Millisekunden-Bereich.

Dieses Konzept ist im Sinne der Toleranz gegenüber Störsignalen und im Sinne des Ruhestrombedarfs optimal, jedoch erfordert die Validierung wie im vorherigen Abschnitt beschrieben weitgehende Modifikationen des Netzwerktransceivers. Die reine Signalerkennung ist einfacher zu realisieren, und eine Wake-Up-Validierung durch das Netzwerkmanagement sichert gegen falsche Weckereignisse ohnehin ab.

Out-of-Band Signalisierung

Eine *Out-of-Band Signalisierung* moduliert über Frequenzmultiplex-Verfahren ein Wecksignal auf die Netzwerkleitung, welches in einem anderen Frequenzband liegt als das Netzwerksignal selbst. Im Prinzip handelt es sich dabei um eine Variante der Weckleitung, bei der das Netzwerkmedium mitbenutzt wird. Notwendig ist hierbei eine Zusatzbeschaltung auf beiden Seiten der Netzwerkleitung, über die ein Wecksignal aufgekoppelt und wieder ausgekoppelt wird. Der einfachste Fall hierfür ist beispielsweise ein Gleichspannungssignal, welches induktiv und kapazitiv auf das Adernpaar gekoppelt wird. Eine komplexere Variante stellt ein frequenz- oder amplitudenmoduliertes Signal außerhalb des Spektrums des Nutzsignals dar, mit dem sich auch einfache Informationen codieren lassen, um eine Validierung – beispielsweise anhand einer Kennung – zu ermöglichen. Nachteilig ist hierbei die notwendige Zusatzbeschaltung, die für ein frequenzmoduliertes Signal mit Codierung vergleichsweise aufwändig wäre und zusätzlich Impedanz und Symmetrie der Netzwerkleitung beeinflussen kann. Ein solcher Ansatz wird in [74] vorgestellt und diskutiert.

Tabelle 5.4: Bewertung der Out-of-Band-Signalisierung

Kriterium	Bewertung
Integrationsaufwand	Hoch, Koppelnetzwerk notwendig.
Validierbarkeit	Bei Verwendung komplexerer Modulations- und Codierverfahren, bedeutet jedoch höheren Integrationsaufwand.
Intrusivität	Keine Veränderung des Netzwerksignals nötig, jedoch mögliche Beeinflussung der elektrischen Eigenschaften der Leitung.
Ruhestrombedarf	Kein Ruhestrom notwendig.
Weckzeit	Schnelle Erkennung möglich (Mikrosekunden).

Tabelle 5.5: Vergleich der Konzepte zum Wecken durch Leitungssignalerkennung

Kriterium	Erkennung	Validierung	Zweistufig	Out-of-Band
Integrationsaufwand	+	-	-	o
Validierbarkeit	-	+	+	o
Intrusivität	+	o	o	-
Ruhestrombedarf	+	-	+	+
Weckzeit	+	-	-	+

Vergleich

In Tabelle 5.5 sind nochmals die Eigenschaften der vorgestellten Konzepte zum Wecken über Leitungssignalerkennung zusammengefasst. Die in den jeweiligen Abschnitten vorgestellten Kriterien werden dabei im Vergleich bewertet, wobei '+' für eine vorteilhafte Ausprägung des Kriteriums, '-' für einen Nachteil, und 'o' für eine ausgewogene Bewertung steht, bei der sich Vor- und Nachteile die Waage halten.

In Summe überwiegen die Vorteile eines Weckens durch eine reine Erkennung von Leitungssignalen, insbesondere unter der Bedingung, dass eine Validierung des Weckereignisses auch auf höheren Schichten, etwa im Netzwerkmanagement erfolgen kann. Deshalb wurde eine Realisierung dieses Konzeptes weiter verfolgt und in Kapitel 6 im Detail beschrieben.

5.4 Überlegungen zur Koordination der Knoten und Teilnetze

Die Koordination des gewünschten Zustandes der Knoten oder Teilnetze über das Netzwerk wurde bereits als ein wichtiger Aspekt von Teilnetzbetrieb genannt. In Abschnitt 4.3 wurde diese Koordination als Teilaufgabe des Netzwerkmanagements dargestellt, und deren Umsetzung im AUTOSAR-Standard für CAN-Busse im Bordnetzwerk beschrieben.

Technologische Unterschiede zu CAN, wie etwa die Punkt-zu-Punkt-Topologie, dedizierte Verbindungen zwischen Knoten und die Weiterleitung von Ethernet-Rahmen über Switches, erfordern ein anderes Konzept zur Koordination als das in AUTOSAR vorhandene.

5.4.1 Wer weckt wen?

Die unterschiedliche Netzwerktopologie von Ethernet und CAN wurde bereits in Abschnitt 5.2 angeschnitten. Während bei CAN alle Knoten über ein gemeinsames Medium miteinander verbunden sind, gibt es bei Ethernet-Netzen Punkt-zu-Punkt-Verbindungen (Links) zwischen den Knoten. Die Endknoten sind sternförmig über Switches miteinander verbunden, die die Ethernet-Frames abhängig von deren Ziel-MAC-Adresse an die jeweiligen Endknoten (oder an weitere Switches) weiterleiten. Broadcast-Frames und Frames, deren Zieladresse einem Switch unbekannt ist, werden von ihm auf allen Ports zu den jeweiligen Nachbarn weitergeleitet. Die Weiterleitung erfolgt jedoch generell nur über Ports, an denen eine aktive Verbindung (Link) zum jeweiligen Nachbarn besteht. Dementsprechend werden Frames an abgeschaltete Nachbarn (deren Netzwerkschnittstelle inaktiv ist) gar nicht erst zugestellt. Die sich daraus ergebenden Implikationen auf das Aufwecken wurden im letzten Abschnitt bereits dargestellt. Das Wecken kann nicht mehr unmittelbar lokal durch Erkennung eines Netzwerkpaketes erfolgen, sondern nur noch durch einen aktiven Nachbarn. Das Konzept hinter dem AUTOSAR-Netzwerkmanagement, bei dem jeder Knoten aufgrund des in den NM-Botschaften mitgesendeten Bitvektors selbstständig aufwacht, lässt sich also nicht ohne Weiteres auf Ethernet übertragen. Vielmehr verlagert sich die Verantwortung auf die Nachbarknoten, die ihre Linkpartner aufwecken müssen. Zur Koordination von Ethernet-Teilnetzbetrieb stellt sich die Frage, welcher Knoten welchen Nachbarknoten wann aufweckt.

5.4.2 Zentrales und dezentrales Netzwerkmanagement

Eine zentrale Koordination vereinfacht den Umgang mit topologischen Gegebenheiten im Netzwerk. Beim zentralen Netzwerkmanagement gibt es einen Netzwerkmanager (Master) der alle anderen Geräte im Netzwerk (Agenten oder Slaves) verwaltet. Der Vorteil hierbei ist, dass wenig bis keine NM-Logik auf den verwalteten Geräten (Switches und Endknoten) notwendig ist und das eingesetzte Protokoll einfach gehalten werden kann. Allerdings zieht dies die Notwendigkeit nach sich, dass der Master-Knoten ein umfassendes und vollständiges Wissen über die gesamte Konfiguration (welche Geräte sind verbaut, wie lauten deren Adressen), Topologie (welches sind die Verbindungen zwischen den Geräten) und den Status (welche Geräte sind gerade aktiv, welche Verbindungen sind aktiv) des Netzwerks besitzen und aktuell halten muss. Ein Nachteil dabei ist, dass jede Änderung der Topologie (Ausstattungsvarianten, Erweiterungen)

im Masterknoten konfiguriert werden muss. Protokolle zur Topologieerkennung können zwar eingesetzt werden, es würde jedoch viel zu lange dauern, die Topologie bei jedem Systemstart erneut zu lernen. Solche Protokolle könnten aber beispielsweise bei der Erstinbetriebnahme des Fahrzeugs oder in der Werkstatt nach Service-Arbeiten ausgeführt werden. Ein weiterer Nachteil liegt darin, dass Knoten andere Knoten nur über eine Anfrage beim Master wecken können, was eine zusätzliche Zeitverzögerung mit sich bringt. Und nicht zuletzt stellt sich die Frage: Wer weckt den Master? Ein Ausfall des Masterknotens selbst oder der Verbindung zum Masterknoten würde zudem zum Ausfall des ganzen Netzwerkmanagements führen. Werden die Knoten wie bei AUTOSAR durch das NM wachgehalten, könnte dies beim Ausfall des Masters sogar dazu führen, dass das gesamte Netzwerk einschläft.

Beim dezentralen Netzwerkmanagement ist auf jedem Knoten ein (gleichberechtigter) Netzwerkmanagement-Agent implementiert, der mit den anderen Agenten über ein Protokoll kommuniziert. Die NM-Logik ist hier auf das gesamte Netzwerk verteilt, es gibt keinen einzelnen Master. Bezogen auf Teilnetzbetrieb bedeutet dies, dass jeder Knoten aufgrund der Informationen, die er von anderen erhält, selbst entscheidet, welchen Nachbarn er wann aufweckt. Um einen anderen Knoten oder ein Teilnetz anzufordern, muss ein Agent den anderen diese Information über das Protokoll mitteilen. Der Vorteil dabei ist, dass kein globales Wissen über das gesamte Netzwerk notwendig ist. Jeder Agent benötigt nur lokales Wissen über seine Umgebung. Damit können sich Teile des Netzes unabhängig von anderen Teilen selbst organisieren, was eine höhere Flexibilität gegenüber unterschiedlichen Netzwerkkonfigurationen und ein einfacheres Haushalten von Zustandsdaten ermöglicht. Der Preis hierfür ist, dass in jedem Knoten mehr Software notwendig ist, sowie mehr benötigte Bandbreite im Netzwerk für die Kommunikation der Agenten. Bei einem Bandbreitensprung von 1 Mbit/s auf 100 Mbit/s, wie es in der Automobilvernetzung der Fall ist, dürfte aber die durch das NM verursachte Netzwerklast keine große Rolle spielen.

Ein hierarchisches Netzwerkmanagement stellt eine Mischform aus zentralem und dezentralem Ansatz dar. Es lassen sich einzelne Teilnetze zusammenfassen, die jeweils zentral von einem Master verwaltet werden. Die Master der einzelnen Teilnetze organisieren sich untereinander dezentral. Auch hier ist kein globales Wissen vonnöten. Jeder Master muss nur über sein eigenes Teilnetz vollständiges Wissen besitzen und seine Nachbar-Master kennen. Die Slaves der einzelnen Teilnetze benötigen überhaupt kein Wissen über das Netzwerk. Dies birgt den Vorteil, dass nicht auf jedem Knoten der volle NM-Agent implementiert sein muss, und somit nicht jedes Gerät höheren

Anforderungen an Rechenleistung und Speicher genügen muss. Gleichzeitig muss aber auch nicht das gesamte Wissen zentral konfiguriert werden und es gibt keinen „Single Point of Failure“.

Alle Ansätze haben ihre Vor- und Nachteile. Im Rahmen dieser Arbeit wurde zunächst ein Ansatz für zentrales Netzwerkmanagement auf Basis des *Simple Network Management Protocol* (SNMP) umgesetzt (vgl. Kapitel 7). Mit *Multinet* (vgl. Kapitel 8) wurde anschließend ein Netzwerkmanagement aufgebaut, welches mehr auf den Einsatz in einer Steuergerätearchitektur zugeschnitten ist und hierarchisch oder vollständig dezentral eingesetzt werden kann.

6 Energy-Detect-Modul: Selektives Wecken über Ethernet

Selektives Aufwecken von Steuergeräten wurde bereits in Abschnitt 5.1 als eine wichtige Voraussetzung von Teilnetzbetrieb genannt. In dem Zusammenhang wurden unterschiedliche Weckverfahren beschrieben, bewertet und verglichen. Im Fazit überwiegen die Vorteile des Weckens durch die Erkennung von Aktivität auf der Leitung, wenn man Aufgaben wie Validierung und Fehlerbehandlung in der (ohnehin notwendigen) Schicht des Netzwerkmanagements beheimatet sieht. Für Ethernet-basierte Netze gibt es noch keine Realisierung zum Wecken über die Netzwerkleitung, die nur geringen Ruhestrom aufweist und Leitungssignale erkennt und als Weckereignis nutzt. In diesem Kapitel wird ein Schaltungsmodul, das diese Aufgabe erfüllt, vorgestellt und beschrieben. Dabei handelt es sich um einen Weckempfänger, der als Schaltungsmodul auf dem Steuergerät sowohl diskret als auch integriert realisierbar ist. Es wird im weiteren Verlauf als Energy-Detect-Modul (EDM) bezeichnet. Im Folgenden werden die Aufgaben und das Funktionsprinzip des EDM beschrieben, und das Schaltungskonzept vorgestellt. Eine Simulation zeigt zunächst das Zeitverhalten des Moduls, insbesondere die prinzipielle Funktionsfähigkeit, die zu erwartende Weckzeit und Erkennungsschwelle sowie deren Abhängigkeiten von der Form des Eingangssignales. Eine prototypische, diskrete Realisierung wurde aufgebaut und vermessen und konnte die Erwartungswerte bestätigen. Das Konzept und die Realisierung des EDM wurden zum Patent angemeldet und veröffentlicht [103]. Die Erkennung von Leitungsaktivität als Weckverfahren für die Umsetzung von Teilnetzbetrieb wurde ebenfalls zum Patent angemeldet [99] und veröffentlicht [96], [95]. Das EDM wurde auch im später beschriebenen Testfahrzeug verwendet.

6.1 Weckempfänger für Basisband-Leitungssignale

Das EDM hat die Aufgabe, ein differenzielles, hochfrequentes (HF) Leitungssignal, wie das *Physical Layer Signaling* von Ethernet, zu erkennen und über ein Gleichspannungs-Ausgangssignal anzuzeigen. Dieses Ausgangssignal kann das Energiemanagement des Steuergerätes in geeigneter Weise aktivieren.

Dabei gelten die folgenden prinzipiellen Anforderungen:

- Geringer Ruhestrombedarf: Das vernetzte Wecken soll so wenig Ruhestrom wie möglich erfordern, um die Batteriestandzeit nicht zu beeinträchtigen (und natürlich um keine Energie zu verschwenden).
- Kurze Weckzeit: Eine möglichst kurze Zeitverzögerung zwischen dem Auftreten des Weckereignisses und dessen Signalisierung ist wichtig, um den gesamten Weckvorgang des Steuergerätes nicht wesentlich zu verlängern. Für viele Systeme im Auto ist eine kurze Weckzeit erforderlich, insbesondere bei Teilnetzbetrieb, da das jeweilige System ansonsten eine hohe Reaktionszeit aufweist. Dies ist nicht nur beim Wecken des Fahrzeugs (auch ohne Teilnetzbetrieb) kritisch, sondern auch bei Systemen die bei Bedarf aktiviert werden und dann ihre Funktion sofort ausführen sollen.
- Geringe Störanfälligkeit: Die elektrische Umgebung der Fahrzeugelektronik ist durch eine Vielzahl an Störsignalen verunreinigt, teils von außen (Mobiltelefone, Radaranlagen, Sendemasten), teils von der Abstrahlung der Elektronik selbst und teils durch elektromagnetische Störungen durch Motoren oder elektrische Lastschwankungen. Ein Weckempfänger sollte auf elektromagnetische Störeinflüsse unempfindlich reagieren, da ansonsten eine hohe Zahl Falscherkennungen und damit ungewollter Weckvorgänge auftritt.
- Keine Beeinflussung des Leitungssignals: Die Integrität des Netzwerksignals darf nicht durch den Weckempfänger beeinflusst werden. Eine Veränderung des Signals hätte eine Verschlechterung der Störfestigkeit des Netzwerks zur Folge, bis hin zu einem völligen Verlust der Kommunikation.

Diese Anforderungen stehen teilweise durchaus im Widerspruch zueinander. Gäbe es beispielsweise den Wunsch nach geringem Ruhestrombedarf nicht, könnte *Wake-on-LAN* ohne Weiteres eingesetzt werden, wobei sowohl die kurze Weckzeit (eine Framelänge) als auch die geringe Störanfälligkeit gegeben wären. Eine kurze Weckzeit

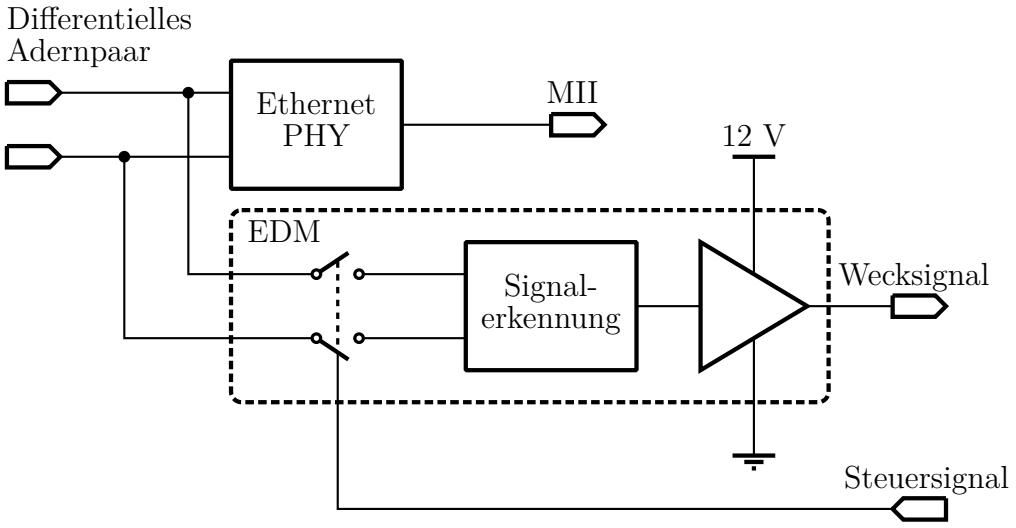


Abbildung 6.1: Schematischer Aufbau des EDM mit schaltbarem Weckempfänger

erschwert wiederum eine Validierung des Weckereignisses und verschlechtert damit die Störsicherheit.

Bei der Auslegung des EDM musste demnach ein Konzept gefunden werden, bei dem alle Kriterien in ausreichendem Maße erfüllt werden. Das Ergebnis ist ein Hochfrequenz-Weckempfänger, der parallel zum Netzwerktransceiver geschaltet ist, einen differentiellen Eingang besitzt und nach erfolgtem Aufwecken von dem differentiellen Leitungspaar abgekoppelt werden kann, um das Leitungssignal in keinster Weise zu beeinflussen. Der Weckempfänger benötigt im weckfähigen Zustand keinen Ruhestrom, die notwendige Energie zur Erkennung wird direkt aus dem HF-Leitungssignal gewonnen. Das grundlegende Prinzip wird in Abbildung 6.1 veranschaulicht.

6.2 Eigenschaften des Ethernet-Leitungssignals

Der Weckempfänger des EDM zeigt das Vorhandensein eines hochfrequenten Wechselsignals auf dem Ethernet-Medium an. Dabei ist seine Charakteristik auf die Eigenschaften des Leitungssignals von Ethernet abgestimmt, das es zu erkennen gilt. Da der Empfänger rein analog arbeitet, spielt dabei die Kanal- und Leitungscodierung des Netzwerksignals keine Rolle. Deswegen ist das EDM auch universell für verschiedene Netzwerkstandards einsetzbar, solange sich das elektrische Leitungssignal grundsätzlich ähnelt. Das Leitungssignal aller Ethernet-Varianten wird im Ethernet-PHY (*Physical Layer Transceiver*) aus den digitalen Daten der MAC-Schicht (*Medium Access*

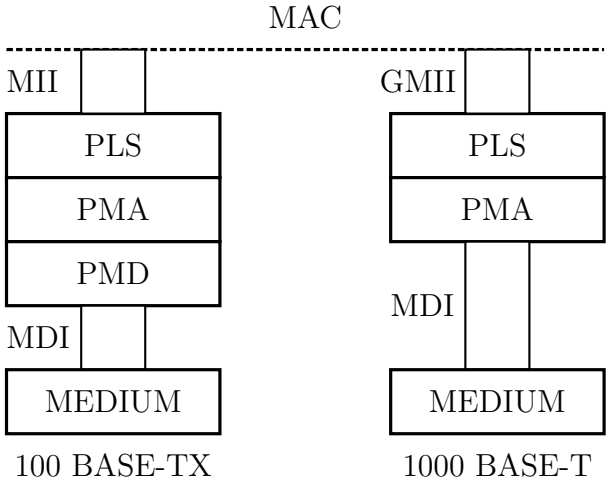


Abbildung 6.2: Bitübertragungsschicht von 1000BASE-T und 100BASE-TX [40]

Control) erzeugt. Abbildung 6.2 zeigt die Subschichten des PHY nach IEEE 802.3 [40]. Sie werden im Folgenden kurz gemäß den detaillierteren Ausführungen in [40] beschrieben.

Der *Physical Coding Sublayer* (PCS) stellt die oberste Schicht und die Schnittstelle zum Medium Independent Interface dar. Die vom MAC zur Übertragung empfangenen Daten werden im PCS in die auf den jeweiligen Leitungen zu übertragenden Symbole codiert, gescramblet und serialisiert. Das Scrambling dient der Optimierung des Signalspektrums auf dem Übertragungsmedium.

Die Schicht *Physical Medium Attachment* (PMA) bildet die Codebits des PCS auf verschiedenen physischen Medien ab. Bei 100BASE-TX stellt sie die Schnittstelle zwischen dem PCS und verschiedenen *Physical Medium Dependent* (PMD) -Schichten dar. Bei 1000BASE-T erzeugt sie direkt die Leitungscodes, die über das *Medium Dependent Interface* (MDI) versendet werden.

Die *Physical Medium Dependent* (PMD) -Schicht beschreibt bei 100BASE-TX die vom Medium abhängigen Eigenschaften des Transceivers wie Leitungscode, Bittiming, Signalform und die elektrischen Eigenschaften des Signals. Bei 1000BASE-T werden diese Eigenschaften bereits im PMA abgedeckt und es gibt keine PMD-Schicht. Jede Schicht verändert den zu übertragenden Datenstrom entsprechend ihren Aufgaben. Daraus und aus den elektrischen Eigenschaften von PMA und PMD ergeben sich die für die Erkennung des Leitungssignals relevanten Charakteristika. Dies sind insbesondere die Signalamplitude, Grundfrequenz und Frequenzspektrum.

6.2.1 100BASE-TX Fast Ethernet

Der PMD von 100BASE-TX wird durch ANSI X3.263-1995 [3] beschrieben, auf den in IEEE 802.3 (Absatz 25.2) referenziert wird [40]. 100BASE-TX *Fast Ethernet* überträgt Daten mit einer Datenrate von 100 Mb/s über ein verdrilltes Adernpaar[1]. Die vom PMA kommenden Symbole werden im PMD 4B/5B NRZ kodiert. Die Symbolrate des Signals liegt daher bei 125 MBaud. Die NRZ-Symbole werden gescrambled und anschließend im PMD-Encoder leitungscodiert. Als Leitungscode wird *Multi-Level-Transitioning-3* (MLT-3) verwendet. Der Signalwert von MLT-3 verändert sich bei aufeinanderfolgenden '1'-Symbolen gemäß der Folge $0, +1, 0, -1, 0$ und bleibt bei '0'-Symbolen unverändert [3]. Näherungsweise ähnelt diese Folge einer Sinus-Periode, während der vier Symbole übertragen werden. Durch MLT-3 wird die spektrale Energie des Signals in niedrigere Frequenzbereiche verschoben und die Grundfrequenz des 125 MBaud-Signals auf ein Viertel, also 31,25 MHz reduziert[2]. Diese MLT-3-Symbole werden schließlich durch den Ausgangstreiber gemäß der in ANSI X3.263 Absatz 9.1 (*Active Output Interface*) standardisierten Signalform auf das differenzielle Leitungspaar gesendet. Die maximale Amplitude des Differenzsignals liegt zwischen 1165 mV und 1285 mV bei geschirmten Leitungen und zwischen 950 mV und 1050 mV bei ungeschirmten. Die Signalanstiegszeit liegt gemäß TP-PMD zwischen 3 ns und 5 ns, die Symbolperiode bei 8 ns. Abbildung 6.3 zeigt eine schematische Darstellung des Leitungssignals.

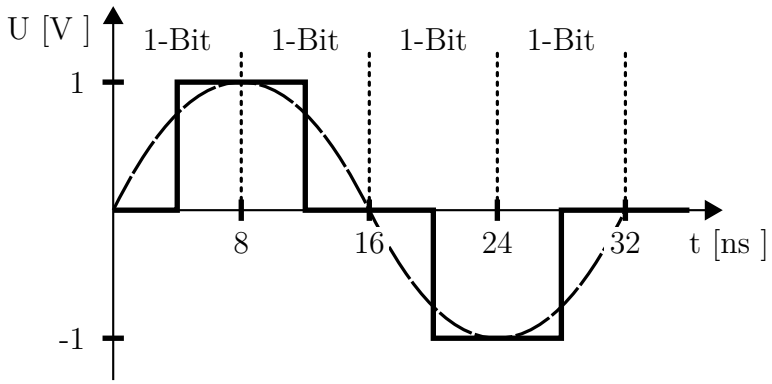


Abbildung 6.3: Darstellung des MLT-3 Leitungssignals von 100BASE-TX

[1]Da 100BASE-TX Vollduplexübertragung bietet, werden insgesamt zwei Adernpaare, eins pro Richtung, genutzt

[2]Zum Vergleich: bei einem NRZ-kodierten Signal werden pro Periode der Grundfrequenz zwei Bit übertragen, also liegt die Grundfrequenz des Signals bei der halben Symbolrate.

6.2.2 1000BASE-T Gigabit Ethernet

Die Eigenschaften des Leitungssignals von Gigabit-Ethernet werden durch IEEE 802.3 Satz 40 spezifiziert, der PCS und PMA für 1000BASE-T beschreibt [40]. Dabei werden 1000 Mb/s über vier verdrillte Adernpaare übertragen, also 250 Mbit/s pro Adernpaar[3]. Die Symbolrate pro Adernpaar liegt bei 125 MBaud, wie auch bei 100BASE-TX. Die doppelte Bitrate bei gleicher Symbolrate liegt in der fünfwertigen Codierung (PAM5, Pulsamplitudenmodulation mit fünf Werten) begründet. Damit können zwei Bit pro Symbol übertragen werden, im Gegensatz zu NRZ und MLT-3, die ein Bit pro Symbol und damit 125 Mbit/s (4B/5B codiert) mit 125 MBaud übertragen. Auch bei Gigabit Ethernet liegt damit die Symbolperiode bei 8 ns. Um die gleiche Robustheit wie bei 100BASE-TX mit einem fünfwertigen Leitungscode zu erreichen, ist eine Vorwärtsfehlerkorrektur notwendig [61], weswegen der zu übertragende Datenstrom im PCS Trellis-codiert wird. Die Trellis-codierten PAM5-Symbolfolgen des PCS werden durch einen Signalformer verzerrt um das Signalspektrum an das von 100BASE-TX anzupassen. Durch die PAM5-Modulation und die Signalverformung ist die Signalform nicht so vorhersagbar wie das MLT-3-Signal von Fast Ethernet. Die Spannungsamplitude des Differenzsignals variiert zwischen 500 mV und 1500 mV (vgl. IEEE 802.3 Satz 40.6). Die Grundfrequenz und das Spektrum sind jedoch ähnlich wie bei 100BASE-TX, bedingt durch die gemeinsame Symbolrate von 125 MBaud und die spektralen Anpassungen. Abbildung 6.4 zeigt eine selbst angefertigte Messung des Leitungssignals.

Bei 1000BASE-T wurde aufgrund der Vollduplex-Nutzung jedes Adernpaars die Takterzeugung durch eine Regelschleife über beide Linkpartner realisiert (*Loop Timing*). Einer der beiden Linkpartner ist dabei *Master* und leitet das Transmitter-Timing aus seiner lokalen Uhr ab. Der andere ist *Slave* und nutzt das Leitungssignal um seinen Transmittertakt daraus rückzugewinnen [40]. Beim Aufbau eines Links beginnt also der als *Master* konfigurierte Knoten mit dem Senden von Symbolen [61], was Implikationen auf das Wecken von Nachbarn hat. Der weckende Knoten muss seinen PHY als *Master* konfigurieren, ein geweckter Knoten seinen als *Slave*.

[3] Die Vollduplexfähigkeit bei 1000BASE-T wird durch gleichzeitige Nutzung eines Adernpaars in beide Richtungen mittels *Echo Cancellation* erreicht

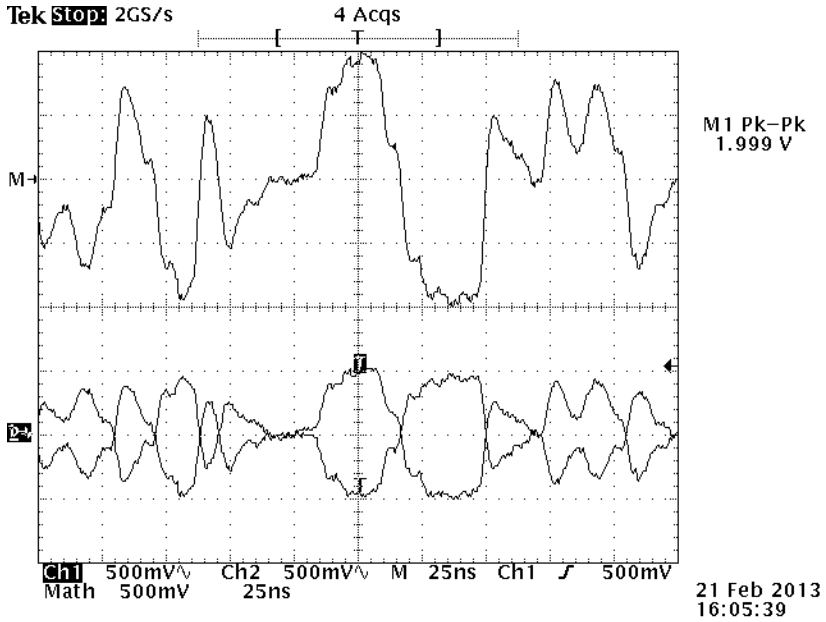


Abbildung 6.4: Aufzeichnung des Leitungssignals von 1000BASE-T Gigabit Ethernet

6.2.3 100BASE-T1 Single Twisted Pair Ethernet

Das unter IEEE 802.3bw [43] spezifizierte 100BASE-T1 (zum Zeitpunkt der Erstellung der Arbeit noch *BroadR-Reach*) basiert auf der Technologie von Gigabit-Ethernet. Die Unterschiede liegen in der Codierung, in der Modulation und in der Signalverarbeitung. In [19] ist die Spezifikation von PLS und PMA zu finden. Grundsätzlich gelten ähnliche elektrische Eigenschaften wie bei 1000BASE-T.

6.2.4 Signalanforderungen für das EDM

Aus den obigen Beschreibungen des Leitungssignals der unterschiedlichen Varianten von Ethernet ergibt sich ein Bild, auf welche Charakteristik der Weckempfänger des EDM ausgelegt werden muss. Die zu erkennenden Signale sind differentiell, besitzen eine Amplitude zwischen 500 mV und 1500 mV und eine Grundfrequenz von ca. 30 MHz bei einer Bandbreite von 125 MHz. Die Signalerkennungsschwelle eines 100BASE-TX PHYs liegt gemäß ANSI X3.263 bei einer Amplitude von 500 mV [3]. Ab dieser sollte deswegen auch das EDM reagieren. Tabelle 6.1 fasst die Signaleigenschaften zusammen.

Tabelle 6.1: Signaleigenschaften für das EDM

Symbol	Beschreibung	Wert
$\hat{U}_{min}$	Minimale Spannungsamplitude	500 mV
$\hat{U}_{max}$	Maximale Spannungsamplitude	2000 mV
f_{min}	Minimale Frequenz	30 MHz
B	Bandbreite	125 MHz

6.3 Aufbau und Schaltungskonzept

6.3.1 Architektur

Der Weckempfänger ist aus drei hintereinandergeschalteten Stufen aufgebaut, die in Abbildung 6.5 dargestellt sind. Die erste Stufe, der HF-Schalter bindet den Weckempfänger an die Netzwerkleitung an und leitet das Signal an die zweite Stufe. Bei der zweiten Stufe handelt es sich um eine Greinacher-Kaskade, die das Wechselstromsignal in eine (höhere) Gleichspannung transformiert. Diese Gleichspannung schaltet die letzte Stufe, den Ausgangstreiber, der das Ausgangssignal zur Indikation einer Erkennung treibt.

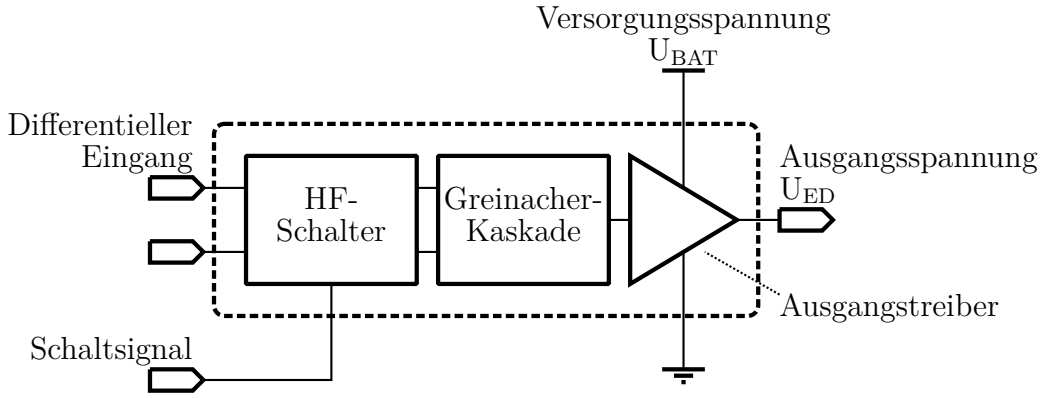


Abbildung 6.5: Blockschaltbild des EDM

6.3.2 Hochfrequenzschalter

Der Aufbau des HF-Schalters ist in Abbildung 6.6 dargestellt. Solange das Steuergerät abgeschaltet ist, bleibt das mit dem Prozessor verbundene Steuersignal U_K hochohmig. Die beiden selbstleitenden MOSFETs (*Metal Oxide Semiconductor Field Effect Transistor*) M1 und M2 sind damit leitend, und der HF-Schalter ist geschlossen. Das EDM

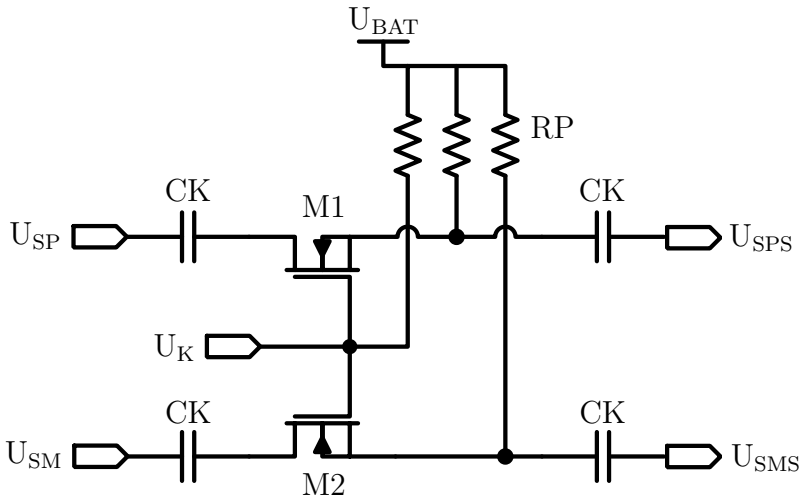


Abbildung 6.6: Hochfrequenzschalter mit n-Kanal MOSFETs

ist damit automatisch an das Medium angekoppelt und scharf. Nach erfolgtem Wake-Up kann der Host-Prozessor das Abkoppelsignal setzen, indem er U_K auf eine Spannung kleiner als $U_{BAT} - U_{GS(p)}$ ($U_{GS(p)}$ bezeichnet die Gate-Source Pinch-Off-Spannung der MOSFETs) treibt. Im einfachsten Fall kann dies durch setzen eines GPIO-Pins (*General Purpose Input/Output*) auf Low-Pegel, also auf Massepotential erfolgen. Die MOSFETs sperren dann und das EDM wird vom Netzwerkmedium hochohmig getrennt.

6.3.3 Greinacher-Kaskade

Das zweite Schaltungsmodul ist eine zweistufige Greinacherkaskade, wie in Abbildung 6.7 dargestellt. Die Greinacherschaltung ist ein Spannungsmultiplikator und Gleichrichter, der ein Wechselstromsignal am Eingang in eine Gleichspannung umwandelt. Die Gleichspannung ist dabei höher als die eingangsseitige Wechselstromamplitude. Unter Vernachlässigung parasitärer Effekte, wie beispielsweise kapazitiver Verluste oder der Vorwärtsspannung der Dioden, ist die Gleichspannung nach jeder Stufe doppelt so hoch wie die Spannung an deren Eingang. Die Ausgangsspannung U_G der Greinacher-Kaskade wäre also unter idealisierten Bedingungen viermal so hoch wie die Signalamplitude am Eingang. Die Anstiegszeit von U_G wird im Wesentlichen durch die Dimensionierung der Kondensatoren CG beeinflusst. Da das Eingangssignal von der Netzwerkleitung differentiell ist, das Ausgangssignal jedoch massebezogen, wird der Eingang durch die beiden Widerstände RS symmetriert. Dadurch wird ein massebezogener Betrieb der Kaskade ermöglicht, während gleichzeitig die Vorteile des

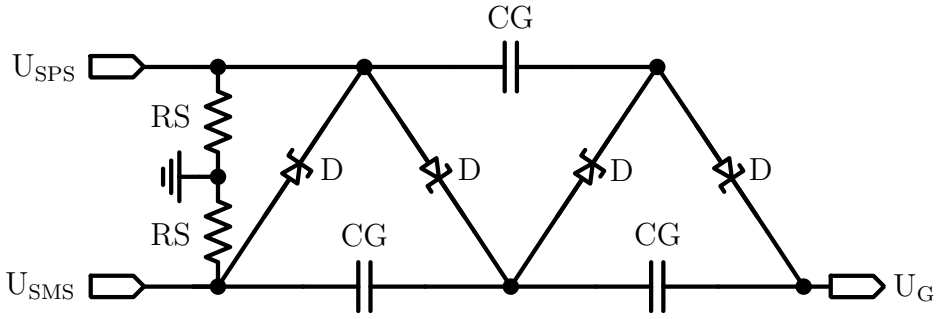


Abbildung 6.7: Zweistufige Greinacher-Kaskade

Differenzsignals erhalten bleiben. Insbesondere Gleichtaktstörungen auf der differentiellen Netzwerkleitung sind damit für die Kaskade nicht sichtbar, wodurch das EDM resistent gegenüber eingekoppelten Spannungen wird.

6.3.4 Ausgangstreiber

Der Ausgangstreiber, dargestellt in Abbildung 6.8, besteht aus zwei invertierenden Verstärkern in Emitterschaltung. Sobald die Ausgangsspannung des Gleichrichters U_G genügend hoch ist, um Transistor T1 zu schalten, wird die Basis von Transistor T2 über Widerstand R4 auf Masse gezogen, und T2 schaltet die Versorgungsspannung U_{BAT} auf den Ausgang U_{ED}. Liegt kein Signal am EDM an, ist U_G null, beide Transistoren sperren, und U_{ED} wird über R5 auf Massepotential gezogen. R1 und C1 bilden einen Tiefpassfilter am Eingang der Treiberstufe, über die hochfrequente Gleichtaktstörungen am Ausgang der Greinacher-Kaskade kurzgeschlossen werden.

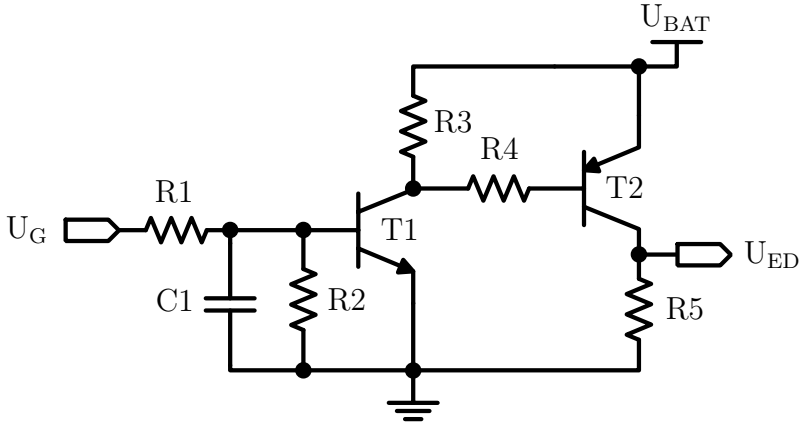


Abbildung 6.8: Schaltung des Ausgangstreibers mit vorgeschaltetem Tiefpassfilter

6.3.5 Gleichtaktunterdrückung

Ein wesentlicher Vorteil differentieller Signale ist ihre Unempfindlichkeit gegenüber Gleichtaktstörungen. Diese betreffen beide Leitungen eines differentiellen Pärchens gleichermaßen, vorausgesetzt die beiden Leitungen sind räumlich dicht beieinander. Bei der Differenzbildung werden Gleichtaktsignale eliminiert, während das Gegentakt-Nutzsignal übrig bleibt. Wie weiter oben bereits angesprochen, ist die Unempfindlichkeit des Weckempfängers gegenüber Gleichtakt-Störungen ein wichtiges Kriterium. Die Greinacher-Kaskade richtet differentielle Wechselstromsignale gleich und multipliziert deren Spannungsamplitude. Gleichtakt-Wechselstromsignale sind für die Greinacher-Kaskade unsichtbar, werden nicht gleichgerichtet und deren Spannung wird nicht hochmultipliziert. Sie tauchen jedoch am (massebezogenen) Ausgang der Kaskade (U_G) als überlagerte Wechselstromsignale wieder auf. Gleichtaktstörungen werden durch die Greinacherstufe nicht verstärkt und gleichgerichtet, sondern unverändert auf den Eingang der Treiberstufe übertragen. Sie sind natürlich unerwünscht, da sie – falls deren Amplitude hoch genug ist – die Ausgangsstufe kurzzeitig schalten könnten, und damit gegebenenfalls eine Fehlerkennung verursachen würden. Ein Tiefpassfilter mit geringer Grenzfrequenz (R1, C1) eliminiert dieses Störsignal. Er lässt nur das gleichgerichtete und verstärkte Nutzsignal auf den Eingang der Treiberstufe durch, während er die Wechselstrom-Anteile der Gleichtaktstörungen abblockt. Der Preis hierfür ist jedoch eine höhere Reaktionszeit des Ausgangs und damit eine größere Signalisierungsverzögerung.

6.4 Charakteristik des Weckempfängers

Der Weckempfänger wurde mit SPICE (*Simulation Program with Integrated Circuit Emphasis*)[4] simuliert, um die Funktion des Moduls zu validieren und auf die Eigenschaften des zu erkennenden Leitungssignals abzustimmen. Die simulierte Schaltung ist mit den entsprechenden Bauteilwerten in Anhang A.2 abgebildet, die zugehörige SPICE-Netzliste in Anhang A.1.

Eine Zeitbereichssimulation wurde mit einem Testsignal nach der Spezifikation aus Tabelle 6.1 in Abschnitt 6.2.4 (vgl. Abbildung 6.9) durchgeführt. Zusätzlich wurde die Simulation mit veränderten Werten der Eingangssignalamplitude $\hat{U}_s$ wiederholt, um

[4]Simulationsprogramm LTSpice

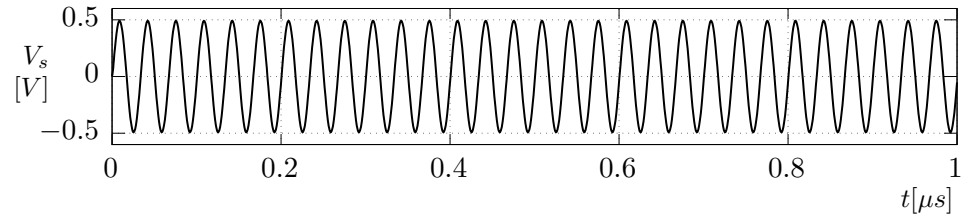


Abbildung 6.9: Eingangssignal mit $\hat{U}_s = 500mV$ und $f_s = 30MHz$

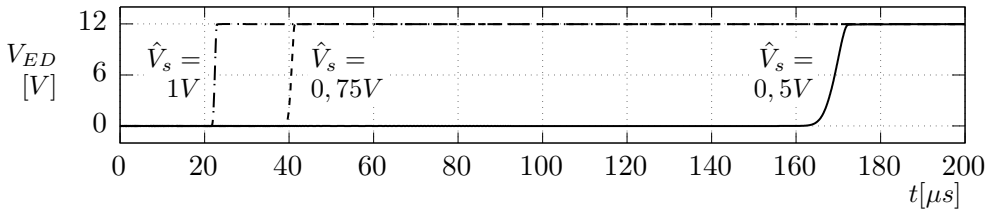


Abbildung 6.10: Ausgang des EDM für versch. Eingangsamplituden $\hat{U}_s$

ihren Einfluss abzuschätzen. Abbildung 6.10 zeigt für drei Werte von $\hat{U}_s$ das Spannungssignal U_{ED} am Ausgang des EDM. Bei der Minimalamplitude von 500 mV schaltet das EDM nach 170 µs. Bei einer Amplitude von 1000 mV beträgt die Schaltzeit ca. 20 µs. Wählt man eine dazwischenliegende Amplitude von 750 mV liegt der Wert sehr nah bei 40 µs. Im schlechtesten Fall (minimale Eingangsamplitude) ist also von einer Schaltzeit von 170 µs auszugehen, in der Regel wird diese jedoch deutlich darunterliegen. Da diese Zeitspannen deutlich unter dem Wert liegen, den ein eingebettetes Gerät zum Hochfahren benötigt (in der Größenordnung 100 ms), ist die durch die Erkennung des Signals zu erwartende Weckverzögerung also vernachlässigbar.

6.5 Systemintegration

Das EDM kann entweder mit diskreten Bauteilen aufgebaut werden, oder als integrierte Schaltung auf einem Microchip. Abbilung 6.11 zeigt den Weckempfänger als Komponente eines eingebetteten Systems, wie beispielsweise eines Steuergerätes. Er wird parallel zum Netzwerk-Transceiver an das differentielle Leitungspaar angeschlossen. Der Abgriff sollte nach den Filterkomponenten der Netzwerkleitungen erfolgen, die bereits Gleichtaktsignale zu einem Großteil filtern. Der Signalisierungsausgang (ED) aktiviert die Spannungsversorgungen des Gerätes, beispielsweise durch Steuerpins der Spannungsregler oder durch Schalten der Batteriespannung. Das Signal wird ODER-verknüpft mit einem Signal des Host-Prozessors (Selbsthaltung), welches nach

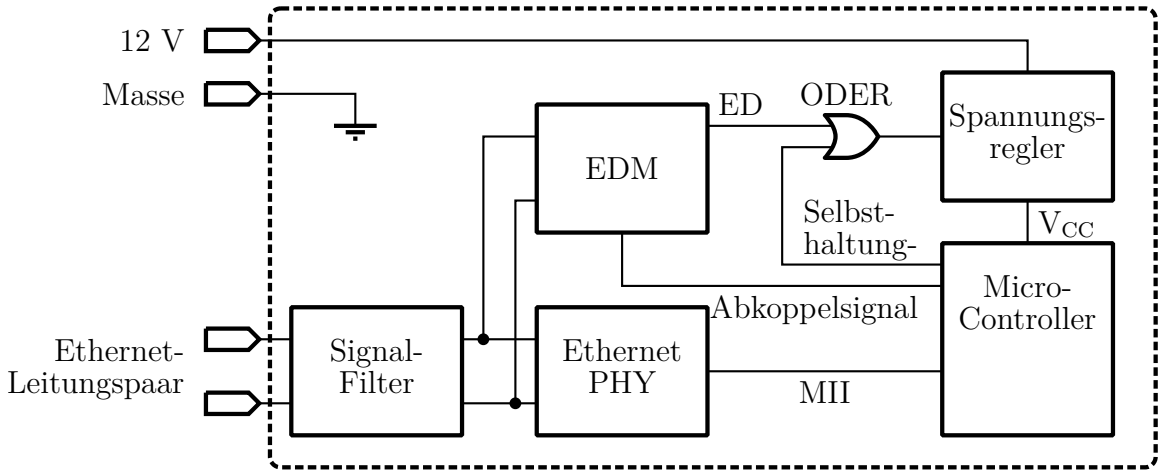


Abbildung 6.11: Blockschaltbild eines Gerätes mit EDM: Der Weckempfänger ist an das differentielle Leitungspaar angeschlossen und aktiviert mit seinem Ausgang die Spannungsversorgung

erfolgtem Wake-Up durch den Prozessor gesetzt wird und die Spannungsversorgung aktiv hält, wenn das Ausgangssignal des EDM wieder verschwindet. Die Selbsthaltung bleibt aktiv, bis der Prozessor das Gerät wieder herunterfährt. Damit ist der Zustand des Steuergerätes nach dem Wecken unabhängig vom Zustand des EDM-Ausgangs. Das Abkoppelsignal wird ebenfalls vom Prozessor nach erfolgtem Start gesetzt, öffnet den Hochfrequenzschalter des EDM und entkoppelt es somit von der Netzwerkleitung.

6.6 Prototypische Realisierung

Das EDM wurde für einen Funktionstest mit realen Netzwerksignalen als Prototyp diskret aufgebaut. Abbildung 6.13 (links) zeigt diesen Prototypen. Das Modul wurde für den Test auf einer eigens entworfenen, einfachen Platine realisiert. Die Anschlüsse (Eingänge, Ausgang, Spannungsversorgung) lassen sich über eine Stiftleiste kontaktieren. Testpunkte zwischen den einzelnen Stufen und bei relevanten Signalen erlauben das messtechnische Erfassen der jeweiligen Spannungsverläufe. Die Bauteilwerte aus der Simulation wurden übernommen.

6.6.1 Test des Prototypen

Der diskrete Prototyp des EDM wurde einem Funktionstest unterzogen, der im Folgenden kurz beschrieben wird. Ziel des Versuchs war, die grundsätzliche Funktionsfähigkeit des Moduls zu überprüfen und das Abkoppeln vom Medium zu validieren. Damit lassen sich vier Teilziele formulieren: 1) Die Erkennung eines 1000BASE-T-Signals und 2) eines 100BASE-T1-*BroadR-Reach*-Signals, 3) die Messung der Signalisierungszeit am Ausgang und 4) ein erfolgreicher Linkaufbau nach Abkoppeln der EDMs. Um die Tests durchzuführen, wurde der in Abbildung 6.12 dargestellte Aufbau verwendet. Zwei Ethernet-Switches wurden miteinander über einen 1000BASE-T-Link (später auch 100BASE-T1 *BroadR-Reach*) verbunden. Der *Slave* wurde mit einem EDM ausgestattet. Dazu wurde am Netzwerkport das erste der vier Adernpaare steckerseitig verzweigt und auf den Eingang eines EDM-Prototypen geführt. Eine kleine Steuerelektronik (S in Abbildung 6.12), über RS-232 mit einem Laptop verbunden, dient dem Schalten der Spannungsversorgung des EDM und des Abkoppel-Signals (DSC) vom Medium. Der ED-Ausgang des EDM ist ebenfalls auf die Steuerelektronik geführt um seinen Zustand durch das Schalten einer LED anzeigen zu können. Auf dem ersten Switch wurde der Port als *Master* konfiguriert, auf dem zweiten als *Slave* und beide Ports wurden abgeschaltet. Zum Ein- und Ausschalten des *Master*-Ports wurde ein Laptop über einen weiteren 1000BASE-T-Link an den Switch angeschlossen und das *Simple Network Management Protocol* (SNMP) verwendet (vgl. Kapitel 7). Auf den zwei Fotos in Abbildung 6.14 ist der Versuchsaufbau dokumentiert. Bei der Durchführung des Versuchs wurde das Netzwerksignal an Messpunkt B (vgl. Abbildung 6.12) aufgezeichnet, und das Ausgangssignal des Moduls an Messpunkt A. Zunächst wurde die

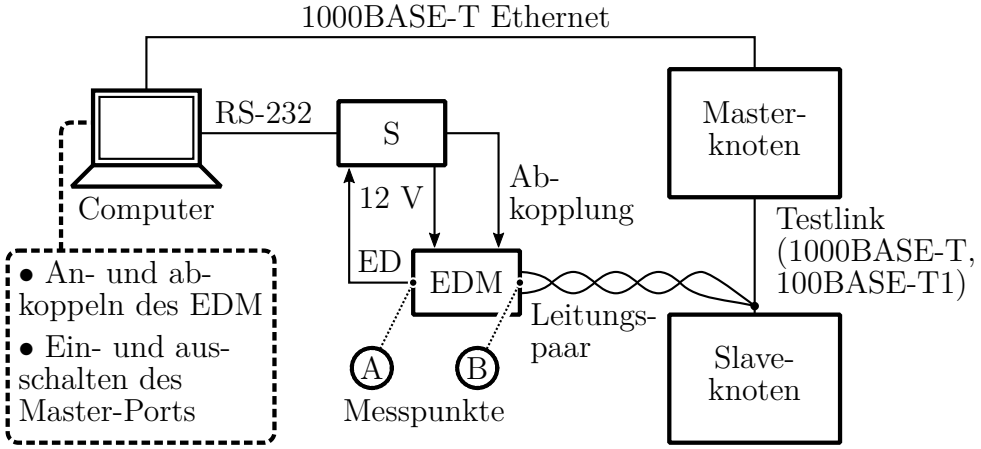


Abbildung 6.12: Schematisch dargestellter Testaufbau

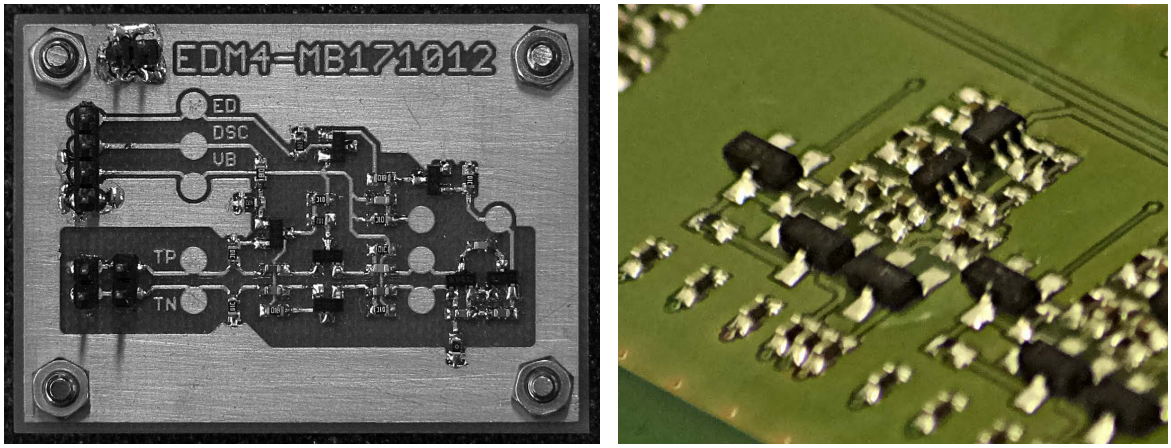


Abbildung 6.13: Diskrete Realisierung als Prototyp (l) und auf Teststeuergerät (r)

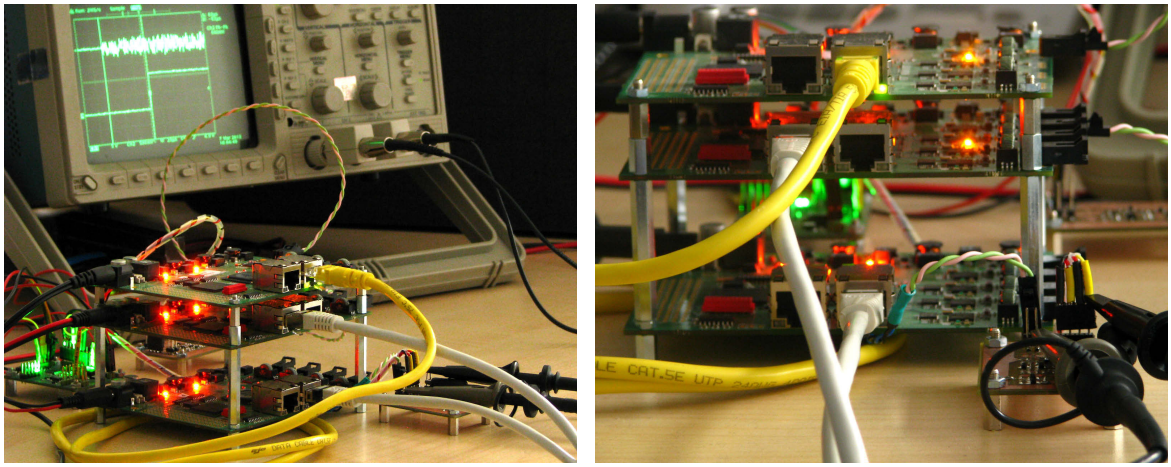
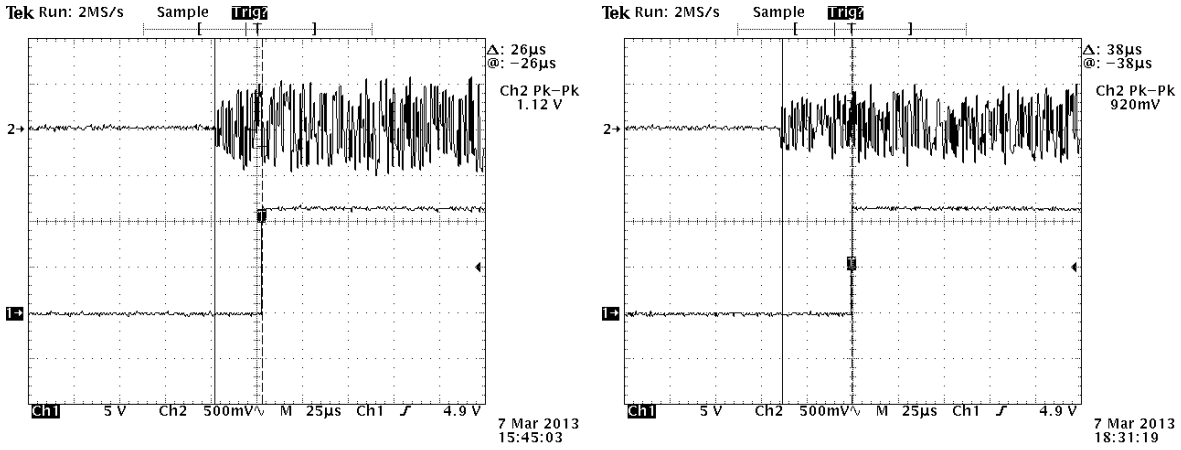


Abbildung 6.14: Aufbau zum Test des EDM-Prototypen



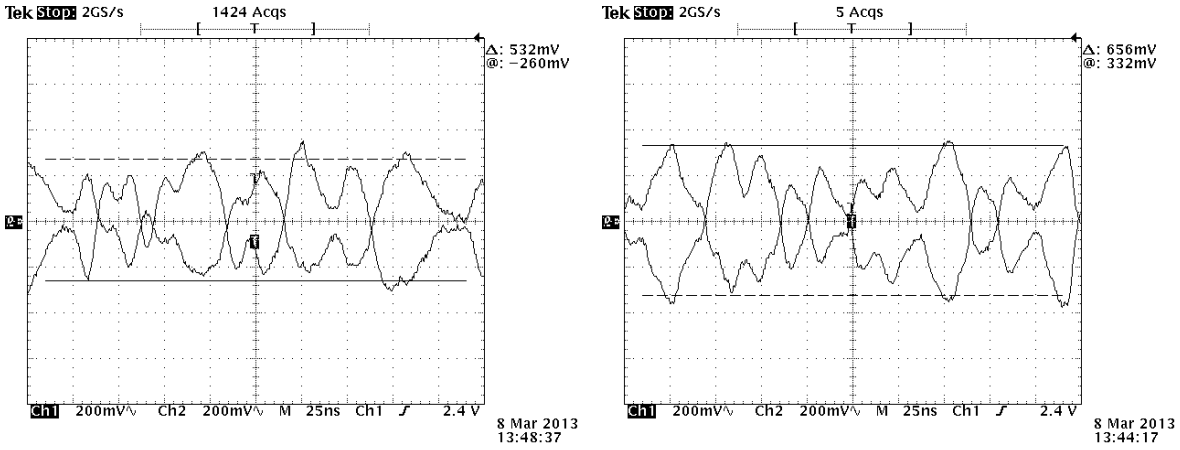
(a) 26 µs, 8 m Leitungslänge, $\hat{V}_s \approx 1100mV$ (b) 38 µs, 70 m Leitungslänge, $\hat{V}_s \approx 900mV$

Abbildung 6.15: Messung der Signalisierungsverzögerung: Kanal 1 zeigt das (positive) Leitungssignal, Kanal 2 den Ausgang des Moduls (Zeitbasis: 1 ms/div)

Erkennung des Signals und die Signalisierungszeit gemessen, indem bei am Medium angeschlossenen EDM der *Master*-Port aktiviert wurde. Anschließend wurde das EDM vom Medium durch setzen des DSC-Signals abgekoppelt, um zu beobachten, ob der Linkaufbau zwischen *Master* und *Slave* erfolgreich ist. Zur Verifikation der Verbindung wurden ICMP (*Internet Control Message Protocol*) *Echo Request*-Pakete an den *Slave gesendet*. Zusätzlich wurde die Signalform bei abgekoppeltem EDM mit der Signalform komplett ohne angeschlossenes EDM verglichen.

6.6.2 Testergebnisse

Das Ergebnis des Versuchs zeigt eine erfolgreiche Erkennung des 1000BASE-T-Leitungssignals bei zwei verschiedenen Leitungslängen (7 m und 80 m). Bei abgeschaltetem *Master*-Port verblieb der ED-Ausgang des Prototypen auf Low-Pegel. Nach Aktivieren des *Master*-Ports wechselte der ED-Ausgang auf High-Pegel. Die Amplitude des Leitungssignals variiert mit der Leitungslänge, und damit (wie auch in der Simulation dargestellt) geringfügig die Signalisierungszeit. In Abbildung 6.15 ist das Messergebnis für beide Kabellängen abgebildet, die Signalisierungsverzögerung liegt im durch die Simulation erwarteten Bereich. Tabelle 6.2 zeigt die gemessenen Werte im Vergleich zu den für die jeweilige Signalamplitude simulierten. Der Prototyp des EDM reagiert geringfügig langsamer als die Simulation erwarten lässt. Dies ist wahrscheinlich darin begründet, dass in der Simulation ein reines Sinussignal verwendet wurde, welches eine höhere Energiedichte besitzt als das Netzwerksignal. Bei angekoppeltem EDM



(a) EDM angekoppelt (DSC=0), 70 m (b) EDM abgekoppelt (DSC=1), 70 m

Abbildung 6.16: Messung des Signalverlaufs beider differentieller Leitungen (Zeitbasis: 25 µs/div)

(DSC nicht gesetzt) kam zwischen *Master* und *Slave* kein Link zustande. Dies ist gut nachvollziehbar, betrachtet man das in Abbildung 6.16a dargestellte Leitungssignal. Die Parallelschaltung von PHY und EDM terminiert jetzt die Leitung, was zum einen zu einem falschen Leitungsabschluss führt. Zum anderen belastet das EDM das Signal asymmetrisch, was für die Signalqualität mindestens ebenso schlimm ist. Nach Abkopplung des EDM durch Setzen des DSC-Pins kommt erfolgreich ein Link zwischen *Master* und *Slave* zustande. Auch das Bild des Leitungssignals sieht jetzt wieder normal aus (vgl. Abbildung 6.16b). Alle Tests wurden auch mit 100BASE-T1 *BroadR-Reach* erfolgreich durchgeführt. Insgesamt wurden somit die durchgeführten Tests des EDM erfolgreich abgeschlossen und eine korrekte, konzeptgemäße Funktion des Moduls validiert. Tests, die eine Anfälligkeit gegenüber Störeinstrahlung abdecken, sind noch nicht durchgeführt worden und Gegenstand weiterer Arbeiten. Beim Einsatz in der Praxis wurden bislang keine Fehlsignalisierungen durch äußere Störungen beobachtet.

Tabelle 6.2: Vergleich der Signalisierungszeit in Simulation und Messung

Signalamplitude	Simulierter Wert	Messwert
$\hat{U}_s = 900mV$	$t_{sig} = 25\mu s$	$t_{sig} = 38\mu s$
$\hat{U}_s = 1100mV$	$t_{sig} = 19\mu s$	$t_{sig} = 26\mu s$

6.7 Einsatz auf einem Teststeuergerät

Das EDM wurde auf einem Testgerät zur Validierung des Gesamtkonzeptes eingesetzt. Das Modul ist in Abbildung 6.13 (rechts) auf der Platine des Gerätes zu sehen. Dies ist genauer in Kapitel 11 beschrieben. An dieser Stelle sei lediglich erwähnt, dass das Modul direkt in die Schaltung des Gerätes integriert wurde und dort aktiv als Weckmechanismus über Ethernet genutzt wird. Mehrere dieser Testgeräte wurden als Netzwerk in einem Fahrzeug integriert, womit das EDM auch seine Funktion beim realen Einsatz im Fahrzeug unter Beweis stellen konnte.

6.8 Wertung

In diesem Kapitel wurde das EDM als Schaltungsmodul zum Wecken von Steuergeräten über Ethernet vorgestellt. Das Grundprinzip des EDM ist die Erkennung des Netzwerksignals auf der Leitung und anschließende Signalisierung über einen elektrischen Ausgang. Dazu wurde ein Weckempfänger entworfen, der das differentielle Netzwerksignal gleichrichtet, verstärkt und damit einen Ausgangstreiber schaltet. Das Ausgangssignal kann die Spannungsversorgung des Host-Gerätes aktivieren und das System somit wecken.

Die eingangs aufgelisteten Kriterien werden erfüllt:

- Geringer Ruhestrombedarf: Das EDM benötigt keinen Ruhestrom während kein Netzwerksignal auf dem Leitungspaar anliegt.
- Kurze Weckzeit: Die Signalisierungsverzögerung des EDM liegt im unteren zweistelligen Mikrosekundenbereich. Das Testsystem wies eine Verzögerung von 20 µs bis 40 µs zwischen Anliegen des Signals und Signalisierung auf. Dieser Wert ist verglichen mit der Startzeit des Gesamtsystems verschwindend klein und verlängert somit die gesamte Weckdauer nur unwesentlich.
- Geringe Störanfälligkeit: Durch Abblocken und Ausfiltern von Gleichtaktsignalen ist das EDM unempfindlich gegenüber Störeinstrahlungen auf der Leitung. Durch Asymmetrien bedingte differentielle Störungen können über die Zeitkonstante ausgefiltert werden, länger anhaltende differentielle Störungen führen zur Fehlerkennung (würden jedoch auch das Leitungssignal stören und sollten daher ohnehin vermieden werden).

- Keine Beeinflussung des Leitungssignals: Das EDM ist mittels HF-Schalter hochohmig von der Signalleitung abkoppelbar und beeinflusst damit die Leitungseigenschaften nicht.

Das EDM ist in dieser Form als Weckmechanismus für das in Kapitel 11 beschriebene Testsystem im Fahrzeug im Einsatz und funktioniert dort zuverlässig und problemlos.

Zum Zeitpunkt der Ausarbeitung wurden die Konzepte und Ergebnisse auch den Halbleiterherstellern, die den Zulieferermarkt für Ethernet im Bereich der Automobilvernetzung abdecken wollten, vorgestellt. Dies war einer der maßgeblichen Faktoren, die zur Integration des Weckmechanismus durch Signalerkennung in die Automotive-PHYs der Hersteller Broadcom und NXP beitrugen. NXP erweiterte das Konzept um mehr Funktionalitäten, wie die Verarbeitung und Weiterleitung von Weckereignissen direkt in Hardware durch die Verwendung spezieller Leitungscodes [78].

Das Schaltungskonzept wurde als Patent angemeldet und veröffentlicht [103]. Das Patent wurde mittlerweile zugeteilt.

7 Zentral gesteuerte Umsetzung von Teilnetzbetrieb

7.1 Teilnetzbetrieb durch zentrales Netzwerkmanagement

Ein zentralisiertes Konzept zur Koordination von Teilnetzbetrieb zielt auf einfache Umsetzbarkeit ab und versucht, die notwendige Informationshaltung und Logik an möglichst einer Stelle zu Bündeln. Im Rahmen der vorliegenden Arbeit wurde ein zentral gesteuertes Konzept zur Umsetzung von Teilnetzbetrieb unter Benutzung des *Simple Network Management Protocols* (SNMP) ausgearbeitet und implementiert. Dies soll im Folgenden kurz zusammengefasst beschrieben werden. Das Konzept wurde zum Patent angemeldet [99] und im Rahmen des Genfer Autosalons 2010 vorgestellt [96]. Die Umsetzung des Konzeptes auf Prototypen-Hardware wurde auch in [88] veröffentlicht.

7.2 Architektur

Die Systemtopologie bei Ethernet ist ein schleifenfreies Punkt-zu-Punkt-Netzwerk. Knoten mit nur einem direkten Nachbarn (bzw. Netzwerklinks) seien Endknoten, Stationen mit mehreren direkten Nachbarn seien Switches. Jede Station kann ihre direkten Nachbarn über ein nachbarbasiertes Weckverfahren aufwecken, wie in Kapitel 5.3 beschrieben. Im Netzwerk gibt es einen Knoten, der die Rolle des zentralen Netzwerkmanagers (ZM) besitzt. Er ist in der Lage, mit den Switches im Netzwerk zu kommunizieren, und fordert bei ihnen portbasiert das gezielte Aufwecken einzelner Nachbarn an.

Damit das System wie beschrieben funktioniert, müssen die folgenden drei Voraussetzungen erfüllt sein:

- Der ZM besitzt ein vollständiges Wissen über die Topologie des Netzes und über die Zustände der Knoten.
- Auf den Switches läuft eine Software (*Switch Agent*, SA), die mit dem ZM über ein geeignetes Protokoll kommunizieren kann, den Zustand der Nachbarknoten überwacht und diese wecken und ferngesteuert herunterfahren kann.
- Jeder Switch weiß, über welchen Port der Pfad zum ZM führt.

Auf diese Weise hat der ZM die vollständige Kontrolle über die Topologie, und kann je nach Bedarf jeden Knoten im Netzwerk aufwecken. Dazu sendet er einen entsprechenden Befehl an den Switch, dessen Nachbar das aufzuweckende Gerät ist. Der SA setzt den Befehl mittels Weckmechanismus um und benachrichtigt den ZM anschließend über die Topologieänderung. Wird ein Knoten durch ein externes Ereignis aufgeweckt, wird durch den entsprechenden Switch ebenfalls eine Mitteilung über die Topologieänderung an den ZM gesendet. Liegen auf dem Pfad zum Ziel inaktive Switches, muss der ZM diese zunächst über ihre jeweiligen Vorgängerknoten wecken. Die Grundlage des Konzeptes wurde in der Diplomarbeit des Autors ausführlich beschrieben [87]. Die Ausarbeitung wurde patentiert und veröffentlicht [99].

Zur Kommunikation zwischen ZM und SAs kann prinzipiell ein proprietäres, einfach gehaltenes Protokoll genutzt werden. Im Rahmen der Konzeptausarbeitung wurde einem eigenen Protokoll das *Simple Network Management Protocol* als Alternative gegenübergestellt. SNMP bietet als standardisiertes Netzwerkmanagementprotokoll alle notwendigen Möglichkeiten. Obendrein lassen sich damit alle anderen Aufgaben des Netzwerkmanagements abbilden, etwa Konfiguration oder Diagnose, so dass Teilnetzbetrieb in ein holistisches Netzwerkmanagement-Konzept integriert werden kann und somit wenig zusätzlichen Aufwand verursacht.

7.2.1 Simple Network Management Protocol

SNMP ist ein Netzwerkmanagementprotokoll aus der Internet-Protokollfamilie, das die Überwachung, Konfiguration und Diagnose von Netzwerkgeräten von einer zentralen Managementstation aus ermöglicht. Die zu verwaltenden Geräte besitzen eine Agentensoftware, mit der die Managementsoftware über das Protokoll SNMP kommunizieren kann. Über einen einfachen Befehlssatz (GET- und SET-Befehle, Antworten

und Benachrichtigungen) können Daten wie Statusinformationen, Systemparameter oder Konfigurationseinstellungen gelesen und geändert werden. Der Agent verwaltet die jeweiligen Daten in Form von Einträgen in einer Datenbank, und stellt diese dem Manager über SNMP zur Verfügung oder ändert sie auf dessen Kommando hin. SNMP bietet hierfür lediglich das Protokoll und die Infrastruktur. Der Agent ist selbst dafür zuständig, über SNMP geänderte Konfigurationseinstellungen oder Parameter auf dem Netzwerkgerät in Hard- und Software umzusetzen und die in der Datenbank enthaltenen Werte und Statusinformationen aktuell zu halten.

SNMP organisiert die zu verwaltenden Daten (*Managed Objects*) in einem hierarchischen System, der *Management Information Base* (MIB) [68]. Dabei sind verwandte Objekte in MIB-Modulen gruppiert [22], die in einem Baum organisiert sind. Viele MIB-Module der Internet-Familie sind durch die IETF in RFCs definiert, wie beispielsweise die SNMP-MIB [68], die Objekte enthält, die SNMP-spezifische Objekte bereitstellt. Die MIB-II [58, 57] beschreibt die Management-Objekte für TCP/IP-Geräte. Beispiele für Objekte der MIB-II sind Informationen über Netzwerkschnittstellen, Statistiken des Paketverkehrs, Adressen, Routingtabellen und viele weitere protokollspezifische Einträge. Sie ist in Gruppen organisiert, und jede der Gruppen ist durch einen eigenen RFC definiert. Neben den standardisierten MIB-Modulen gibt es auch zahlreiche durch Hersteller von Netzwerkgeräten erstellte MIBs, die spezifische Objekte für die jeweiligen Geräte enthalten. Jedes MIB-Modul ist nach einem festen Regelwerk, der *Structure of Management Information* (SMI) [56] in der Beschreibungssprache ASN.1 (*Abstract Syntax Notation 1*) geschrieben und damit in die SNMP-Infrastruktur einbindbar. So kann mit SNMP und einem passenden MIB-Modul nahezu jedes netzwerkfähige Gerät in beliebiger Weise ferngesteuert, verwaltet und überwacht werden.

SNMP ist in der aktuellen Version 3 durch RFC 3410 [22] und Folgende bis 3418 definiert und der wichtigste Netzwerkmanagement-Standard in der Internet-Protokollfamilie. Die erste Version von SNMP wurde bereits 1990 in RFC 1157 [23] publiziert. 1993 ersetzte SNMPv2 die alte Version [21], und erweiterte das Protokoll um einige Sicherheitsmechanismen.

7.2.2 SNMP als Basis für zentralen Teilnetzbetrieb

Die Struktur von SNMP deckt sich in hohem Maße mit der Struktur des zentralisierten Teilnetzbetriebs und bietet alle notwendigen, infrastrukturellen Mechanismen. Die im

Abschnitt 7.2 beschriebenen Switch-Agenten entsprechen in ihrer Aufgabe den SNMP-Agenten, auf die der zentrale Managementknoten über SNMP zugreifen kann. Der Zustand der direkten Nachbarn eines Switches kann über eine spezifische MIB abgebildet werden. Der Agent muss dann Änderungen der jeweiligen Objekte umsetzen, indem er die entsprechenden Nachbarn aufweckt und den Ist-Zustand als Benachrichtigung an den Manager meldet. Der große Vorteil der Nutzung von SNMP an dieser Stelle gegenüber einem anwendungsspezifischen Protokoll liegt in der Möglichkeit, über SNMP sämtliche anderen Aufgaben des Netzwerkmanagements mit abzudecken. Die Konfiguration von Steuergeräten, die Fehlerüberwachung und die Diagnose ließe sich mit einer auf SNMP basierenden Management-Architektur vollständig abdecken. Das Wecken von Nachbarn ist damit für die Knoten lediglich ein weiteres *Managed Object* in der Geräte-MIB, neben vielen anderen, womit die zentrale Umsetzung von Teilnetzbetrieb als Komponente in einem umfassenden Netzwerkmanagementkonzept integrierbar ist. Der SNMP-Agent stellt damit auf jedem Knoten eine zentrale für alle Aufgaben des Netzwerkmanagements zuständige Softwarekomponente dar.

Über eine spezifische MIB ließen sich alle Konfigurationsparameter und Zustandsinformationen als *Managed Objects* für das Netzwerkmanagement im Automobil abbilden und gruppieren. Eine solche Automotive-MIB zu definieren, liegt jedoch nicht im Fokus der vorliegenden Arbeit. In der im folgenden Abschnitt beschriebenen, prototypischen Implementierung des Konzeptes wurde für das Wecken der Nachbarn ein existierendes Objekt aus der MIB-II benutzt.

7.3 Implementierung

Das Konzept wurde prototypisch auf eingebetteter Hardware umgesetzt. Im Fokus stand dabei die Entwicklung und Integration eines SNMP-Agenten, der unter Benutzung des Energy-Detect-Moduls als Weckmechanismus Nachbarknoten auf Befehl eines Management-Systems aufwecken und abschalten kann.

7.3.1 Systemarchitektur

Abbildung 7.1 zeigt ein einfaches Netzwerk mit zwei SNMP-verwalteten, kaskadierten Switches und jeweils einem damit verbundenen Endknoten. Jedes Gerät ist mit Energy-Detect-Modulen an seinen Netzwerkports ausgestattet, so dass es von seinen Nachbarn

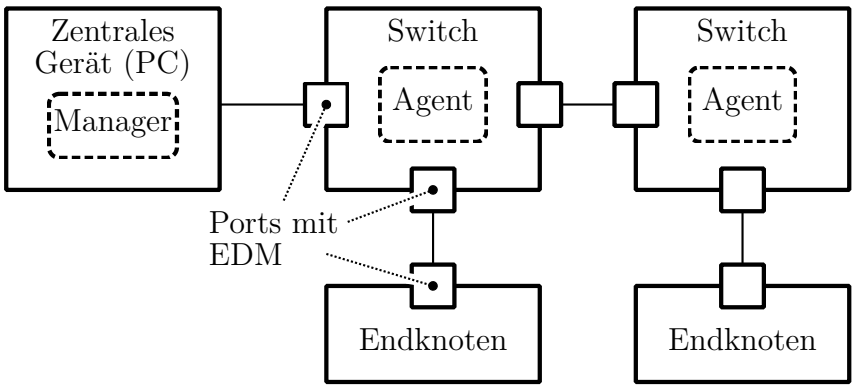
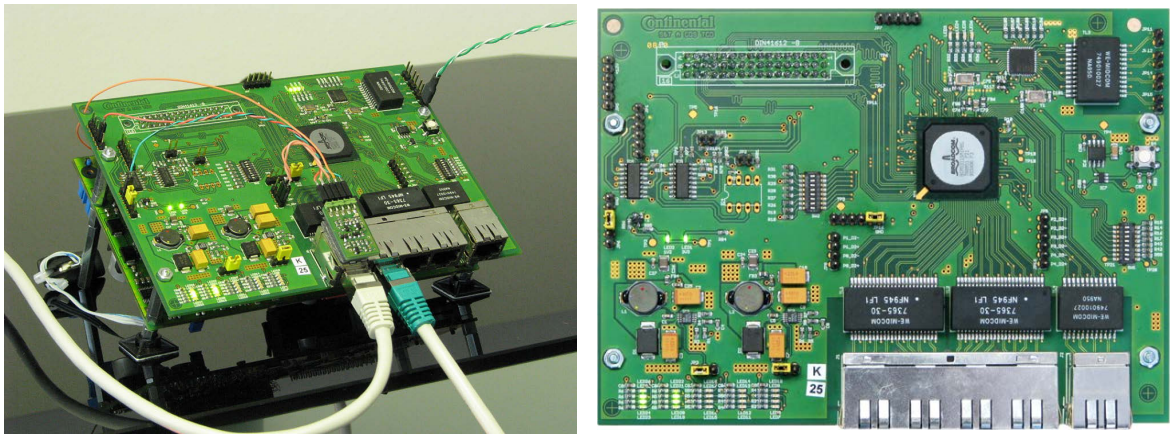


Abbildung 7.1: Zielaufbau mit SNMP-Agenten und weckfähigen Ethernet-Knoten

aufgeweckt werden kann. Der zentrale Manager ist als Softwaremodul auf einem PC realisiert. Dazu eignet sich prinzipiell jedes SNMP-fähige Programm, mit dem auf die *Managed Objects* der Agenten zugegriffen werden kann.

7.3.2 Switch

Die Integration anwendungsspezifischer Software wie dem SNMP-Agenten und des Energy-Detect-Moduls auf einem Ethernet-Switch ließ sich am einfachsten durch den Aufbau einer eigenen Switch-Plattform bewerkstelligen. In der ersten Version wurde der Switch ohne Microcontroller auf einer separaten Platine realisiert und an ein bestehendes Gateway-Steuergerät angebunden. Bei dem Microcontroller des Gateways handelt es sich um einen Freescale 32-Bit PowerPC der Familie MPC55xx, der über einen e200z6-Kern mit 132 MHz Taktfrequenz, 2 MB integrierten Flash-Speicher und 80 kB SRAM verfügt. An Peripherie sind unter anderem ein Fast Ethernet Controller (FEC), sowie mehrere CAN und FlexRay-Controller enthalten. In einer späteren Version wurde ein Microcontroller der MPC56xx-Familie direkt auf der Switch-Platine integriert, so dass das Gerät nur noch aus einer Leiterplatte besteht. Als Switch wurde die integrierte Ethernet-Multiport-Bridge BCM53115M von Broadcom [18] verwendet. Die Bridge verfügt über sieben Ethernet-Ports, davon fünf mit integrierten 10/100/1000BASE-T Ethernet-PHYs. Bei den beiden anderen Ports wird die MII-Schnittstelle (*Media Independent Interface*) des Ethernet-MAC zur Anbindung eines externen PHYs oder eines Prozessors herausgeführt. Der MAC des PowerPC ist über RvMII (*Reverse Media Independent Interface*) mit dem MACs des *In-Band Management Ports* (IMP) der Multiport-Bridge verbunden. Zur Konfiguration und Steuerung der Bridge durch den PowerPC wird der Management-Bus MDIO (*Mana-*



(a) Managed Switch mit Gateway und EDM-Prototyp am ersten Port

(b) Draufsicht auf die bestückte Leiterplatte des Switches

Abbildung 7.2: Erste Version des *Managed Switches*

gement Data Input/Output) der MII-Schnittstelle verwendet. In Abbildung 7.2 ist die erste Version des *Managed Switches* mit rückseitig angeschlossener Gateway-Platine zu sehen. Abbildung 7.2a zeigt das aufgebaute Gerät, an dem ersten Ethernet-Port ist ein Prototyp des Energy-Detect-Moduls angeschlossen. Das Energy-Detect-Modul ist mit dem *Receive*-Leitungspärchen des 100BASE-TX PHYs verbunden und dessen Ausgang mit dem Wecksignal der Spannungsversorgung des Gateways. Erkennt das EDM Link-Aktivität, aktiviert es Gateway und Switch-Platine. Der *Managed Switch* stellt damit die Basis für die Integration des SNMP-Agenten dar. Auf dem Gateway-Controller läuft ein proprietäres, auf AUTOSAR basierendes Betriebssystem, welches die Laufzeitumgebung für den Agenten bietet und Zugriff auf Peripheriekomponenten ermöglicht.

7.3.3 SNMP-Agent

Der Agent stellt das Bindeglied zwischen *Managed Objects* und dem darunterliegenden System dar und erfüllt zwei Aufgaben. Dies sind zum einen die Kommunikation mit dem SNMP-Manager und die Verwaltung der MIB, zum anderen das eigentliche Steuern der Knotenzustände der jeweiligen Nachbarknoten.

Zur Integration von SNMP wurde eine existierende *Open-Source*-Implementierung eingebunden. Dabei wurde die Software *Agent++* [32] verwendet. *SNMP++* und *Agent++* wurden im Rahmen einer durch den Autor betreuten Diplomarbeit [104] auf dem System integriert. Da die Definition einer eigenen MIB den Rahmen dieser Arbeit

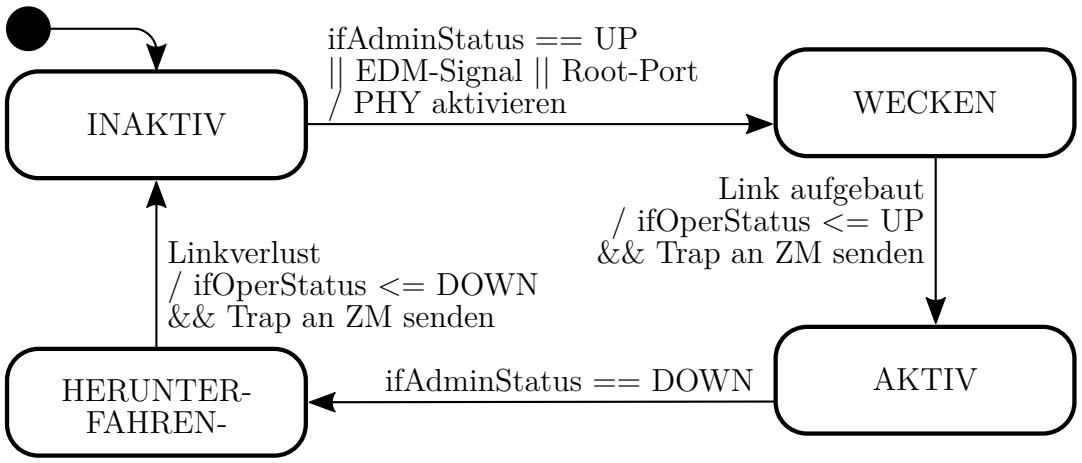


Abbildung 7.3: Vereinfachtes Zustandsdiagramm des Nachbarmanagements

sprengen würde, wurden existierende *Managed Object* aus der MIB-II verwendet. Da das Wecken der Nachbarn mittels EDM portweise durch aktivieren des jeweiligen PHYs erfolgt (vgl. Kapitel 6), wurden die Objekte *ifAdminStatus* und *ifOperStatus* aus der *Interfaces*-Gruppe (vgl. [55]) der MIB-II zweckentfremdet.

- *ifAdminStatus:* Beschreibt den gewünschten Zustand einer Netzwerkschnittstelle im Sinne von 'aktiv' (*up*) oder 'inaktiv' (*down*). In diesem Kontext wird die Bedeutung erweitert auf den gewünschten Zustand des Nachbarknotens und der Netzwerkverbindung zu ihm.
- *ifOperStatus:* Beschreibt den tatsächlichen Zustand der Netzwerkschnittstelle, bzw. hier den Zustand des Nachbarknotens.

Durch Einbinden der MIB-II ließ sich somit das Konzept schnell realisieren ohne ein eigenes, spezifisches MIB-Modul erstellen zu müssen. Dies ist im Detail in [104] beschrieben.

Das Wecken und Schlafenlegen der Nachbarknoten wurde durch einen Zustandsautomaten realisiert. Abbildung 7.3 zeigt das zugehörige Zustandsdiagramm (Anmerkung: Vereinfachte Darstellung, Fehlerzustände wurden weggelassen). Ist ein Nachbar ausgeschaltet steht der Zustand des entsprechenden Netzwerkports auf DOWN. Wird *ifAdminStatus* des Ports über SNMP auf '*up*' gesetzt, wird der PHY eingeschaltet und beginnt mit dem Senden von Leitungssignalen, der Zustand wechselt auf WECKEN. Die Leitungsaktivität weckt den Nachbarn über dessen EDM. Sobald der Netzwerklink zwischen beiden Knoten aufgebaut ist, also der Status des PHYs einen gültigen Link anzeigt, setzt der Agent *ifOperStatus* auf '*up*' und sendet ein SNMP-Trap an den

Manager, um ihn darüber zu informieren, und der Zustand wechselt auf AKTIV. Wird *ifAdminStatus* über SNMP auf '*down*' gesetzt, wird der Nachbar heruntergefahren. Sobald dies geschehen ist, wird *ifOperStatus* ebenfalls auf '*down*' gesetzt, abermals ein SNMP-Trap an den Manager geschickt und der Zustand auf INAKTIV gesetzt. Eine Besonderheit stellt der Root-Port dar. Er wird beim Einschalten sofort aktiviert, um eine Verbindung zum zentralen Manager herzustellen. Wird ein Gerät durch ein externes Ereignis aufgeweckt, wird somit automatisch der gesamte Pfad bis zum Manager aktiviert. Wird ein Knoten durch einen Nachbarn aufgeweckt, wird der jeweilige Port durch Auswerten der EDM-Signale identifiziert und ebenfalls hochgefahren, um die Verbindung mit dem Nachbarn herstellen zu können. Über die SNMP-Agenten lässt sich damit der Zustand der gesamten Topologie vom zentralen Manager aus steuern und mittels Trap-Benachrichtigungen auf Änderungen überwachen.

7.3.4 Zentraler Manager

Der zentrale Manager sendet SNMP-Nachrichten an die Agenten, um Steuergeräte aufzuwecken oder schlafen zu legen. Die Entscheidung, welches Steuergeräte gerade notwendig sind und welche nicht, muss der ZM anhand fahrzeug-, hersteller-, situations- und fahrerspezifischer Zustandsinformationen treffen. Diese Entscheidungslogik ist nicht Teil der vorliegenden Arbeit, vielmehr liegt der Fokus auf der notwendigen Infrastruktur zur Umsetzung von Teilnetzbetrieb und damit auf der Entwicklung der Switch-Agenten. Die Implementierung des ZM wurde daher so einfach wie möglich gehalten, um die Machbarkeit zu zeigen. Sie besteht aus einem SNMP-Backend, mit dessen Hilfe GET- und SET-Botschaften gesendet und TRAP-Ereignisse empfangen werden können, sowie aus einer grafischen Oberfläche, über die Geräte per Mausklick geweckt und schlafen gelegt werden können.

Die SNMP-Anbindung wurde ebenfalls im Rahmen der Diplomarbeit von Benedikt Bartz durchgeführt. Als Implementierung wurde auch hier SNMP++ [32] genutzt. Die Details der Anbindung sind in [104] beschrieben.

7.4 Demonstratoren

Das auf SNMP aufbauende Teilnetzbetrieb-Konzept wurde im Rahmen der vorliegenden Arbeit in mehreren Demonstrator-Aufbauten umgesetzt und vorgeführt. Der in

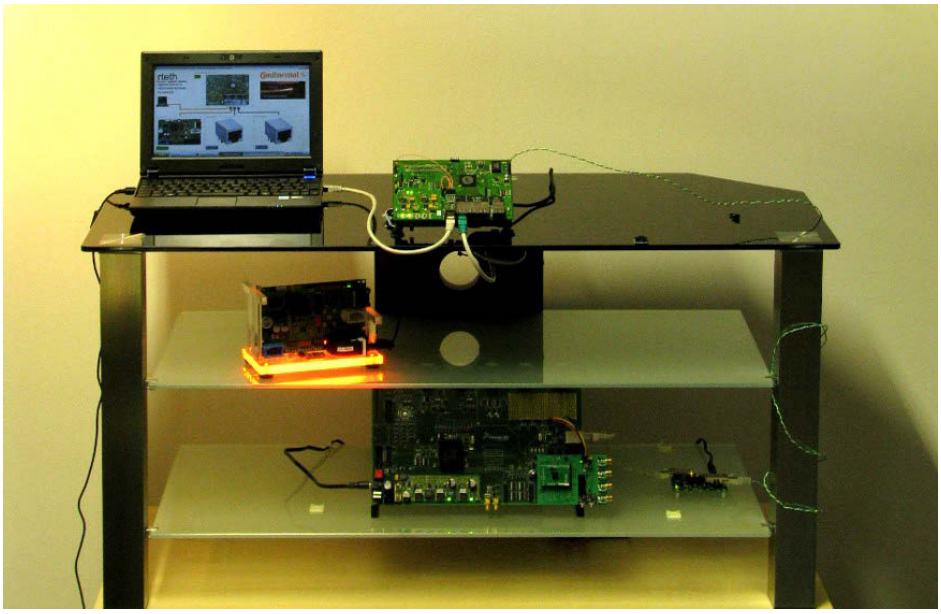


Abbildung 7.4: Tischaufbau mit zentralem Manager (Laptop oben links), *Managed Switch* mit SNMP-Agent (oben mitte), und Ethernet-Steuergerät mit EDM (mitte, beleuchtet)

Abbildung 7.4 gezeigte Tischaufbau zeigte die Funktionsweise mit einem einfachen System aus Manager, Switch mit Agent und angeschlossenem Steuergerät mit EDM. Das Steuergerät konnte über die ZM-Implementierung am Laptop per Mausklick geweckt und ausgeschaltet werden. Später wurde der Aufbau um einen kaskadierten Switch und ein weiteres Steuergerät erweitert. Teilnetzbetrieb über SNMP wurde auch in einem Messeaufbau zusammen mit anderen Aspekten der Ethernet-Vernetzung gezeigt (Abbildung 7.5). Der zentrale Manager wurde auf einem Multimedia-Prototypensystem mit integriert, über das der Demonstrator gesteuert wurde. Für jeden Anwendungsfall des Demonstrators wurden die dazugehörigen Endknoten über das Teilnetzbetrieb-Konzept geweckt und die nicht benötigten Komponenten abgeschaltet. Am zentralen Switch sind ein zweiter Switch, ein digitaler Audio-Verstärker, ein Antennenmodul und das Multimediasystem angeschlossen. Am zweiten Switch sind zwei Ethernet-Kameras angeschlossen, die mittels *Power over Ethernet* (PoE) versorgt und geweckt werden. Der gesamte Demonstrator wurde über CAN durch einen *Body Controller* mittels Funkschlüssel geweckt. Anwendungsfälle des Demonstrators waren Audiostreaming und *Universal Plug and Play* (UPnP), Videostreaming von Ethernet-Kameras und Teilnetzbetrieb.

Die Komponenten waren mit Ausnahme der Kameras, die über PoE geweckt wurden, mit EDMs ausgestattet. Das Multimediasystem, Antennenmodul und der Digitale

Abbildung 7.5: Integrierter Messeaufbau mit Multimedia-System als zentralem Manager, zwei Switches und verschiedenen Endknoten (IP-Kameras, digitaler Verstärker und Antennenmodul)

Verstärker wurden mit einer erweiterten Variante des EDMs, ausgeführt als Vorschaltplatine mit Microcontroller und Schalttransistor, versehen, da nicht auf die Hardware zugegriffen werden konnte. Der Messeaufbau wurde unverändert in ein Demonstrationsfahrzeug eines deutschen Automobilherstellers integriert.

7.5 Zusammenfassung und Diskussion

Das in diesem Kapitel vorgestellte Konzept beschreibt die Umsetzung von Teilnetzbetrieb bei Ethernet-Basierten Netzen mittels zentralem Manager und durch Agenten verwalteten Ethernet-Switches. Der zentrale Manager kennt die Netzwerktopologie, kommuniziert mit den Agenten und lässt sie gezielt ihre direkten Nachbarn aufzuwecken. Als Grundlage für die Netzwerkmanagement-Architektur dient dabei das Protokoll SNMP, über das Manager und Agenten miteinander kommunizieren können. Über Definition spezifischer *Managed Objects* lässt sich der Zustand der jeweiligen Nachbarn eines Agenten in einem MIB-Modul abbilden und über SNMP durch den Manager steuern. Die Umsetzung erfolgt durch den Agenten mittels einer Zustandsmaschine für jeden seiner Ports und direktem Zugriff auf die Hardware. Teilnetzbetrieb lässt sich somit als Komponente eines gesamtheitlichen, auf SNMP basierten

Netzwerkmanagements integrieren. SNMP ist auch für eingebettete Systeme geeignet (es wird beispielsweise auf Netzwerkdruckern, etc. eingesetzt), ausgereift und stellt den wichtigsten Netzwerkmanagement-Standard der Internet-Protokollfamilie dar. Die Idee, das Netzwerkmanagement im Fahrzeug vollständig über SNMP auszuführen hat durch die Modularität und freie Gestaltbarkeit von MIB-Modulen einen gewissen Charme und wurde auch im Rahmen des durchs Bundesministerium für Bildung und Forschung geförderten Kooperationsprojektes SEIS (Sicherheit in eingebetteten, IP-basierten Systemen) veröffentlicht und vorgeschlagen. Sie stellt jedoch einen Bruch mit den üblicherweise im Automobil verwendeten Strukturen, wie etwa dem in Abschnitt 4.3 beschriebenen AUTOSAR-Netzwerkmanagement, dar. Der zentrale Ansatz bringt eine Reduktion der Komplexität der einzelnen Komponenten und einfache Implementierbarkeit mit sich, hat jedoch auch Nachteile.

7.5.1 Robustheit gegenüber Ausfällen und Paketverlusten

Auf der Hand liegt der berüchtigte *Single Point of Failure*. Fällt der zentrale Manager aus, ist das gesamte Netzwerkmanagement außer Funktion. Diagnose, Wecken und Schlafenlegen von Steuergeräten oder Koordination des Zustandes des gesamten Netzwerks sind nicht mehr möglich. Ein weiteres Problem stellen Paketverluste dar, da SNMP über das unzuverlässige Transportschichtprotokoll UDP (*User Datagram Protocol*) transportiert wird. Damit können sowohl SET-Befehle als auch TRAP-Benachrichtigungen verloren gehen, was zu einem inkonsistenten Zustand führt. Bliebe das Netzwerk oder Teile davon durch einen Paketverlust oder einen Fehlerzustand des zentralen Managers fälschlicherweise wach, hätte dies eine Entladung der Fahrzeugbatterie zur Folge, was in jedem Fall vermieden werden muss. Verteilte Ansätze, wie das AUTOSAR-Netzwerkmanagement nutzen daher einen aktiven Wachhaltemechanismus durch zyklische Botschaften von mehreren Knoten. Wenn kein Knoten mehr Wachhalte-Nachrichten sendet, schalten sich die Steuergeräte aus. Dies erhöht die Robustheit des Systems gegenüber Ausfällen einzelner Knoten oder verlorenen Botschaften.

7.5.2 Zentrale Entscheidung über Steuergeräte-Zustände

Auch die zentral gelagerte Entscheidung, wann welches Steuergerät aufwachen bzw. schlafen gehen soll, kann ein Nachteil sein. Sie setzt die umfassende Verfügbarkeit aller

zustandsrelevanten Informationen beim zentralen Manager voraus, auf dessen Basis er Entscheidungen treffen muss. Bei knapp hundert Steuergeräten in heutigen Fahrzeugen kann die Verarbeitung dieser Informationsmenge sehr rechenaufwändig und die Entscheidungslogik komplex werden. Gibt man Steuergeräten selbst die Möglichkeit, Teilnetze (bzw. andere Steuergeräte) zu wecken bzw. anzufordern, kann dies die Komplexität erheblich reduzieren, da die Logik und die notwendigen Informationen auf handhabbare, kleine Einheiten aufgeteilt werden. Jedes Steuergerät kann so aufgrund seiner Informationen selbst entscheiden welche anderen Steuergeräte es zum Erfüllen seiner Funktion benötigt. Dafür bietet SNMP auf Seite der Agenten keine elegante Möglichkeit, so dass das Protokoll modifiziert werden müsste.

7.5.3 Integrierbarkeit in heterogene Fahrzeugarchitektur

Die Integration in ein bestehendes AUTOSAR-Netzwerkmanagement mit CAN-Teilnetzbetrieb ist machbar. Der zentrale Manager muss hierbei die NM-Botschaften der anderen Busse auf auswerten und die entsprechenden Ethernet-Steuergeräte wecken und schlafen legen. Dies erfordert softwareseitig gewissen Aufwand, da periodische NM-Botschaften zu einem virtuellen Topologieabbild angeforderter Steuergeräte zusammengesetzt werden müssen.

7.5.4 Fazit

Das zentrale Netzwerkmanagement im Fahrzeug über SNMP mit Teilnetzbetrieb ist ein vielversprechendes Konzept, insbesondere wenn man von rein IP-basierten Fahrzeugnetzwerken ausgeht und eine zentralisierbare Entscheidungslogik für das Wecken und Schlafenlegen von Steuergeräten voraussetzt. Weitere Arbeit muss in die Robustheit gegenüber Ausfällen und Paketverlusten investiert werden. In der revolutionär geprägten Architektur heutiger Fahrzeuge lässt sich ein rein IP-basiertes, zentrales Netzwerkmanagement auf Basis von SNMP nur schwer realisieren. Im nächsten Kapitel wird *Multinet*, ein dezentrales Protokoll zur Umsetzung von Teilnetzbetrieb vorgestellt, welches die bei AUTOSAR verwendeten Mechanismen, wie zyklische Botschaften, auf Ethernet überträgt. Somit ermöglicht es höhere Robustheit und einfache Integration in bestehende NM-Architekturen. *Multinet* ist vom Grad der Zentralisierung flexibel gestaltbar, so dass dezentraler, hierarchischer und zentraler Teilnetzbetrieb gleichermaßen umsetzbar und kombinierbar sind.

8 Multinet: Eine verteilte Middleware zur Umsetzung von Teilnetzbetrieb

8.1 Übersicht

Multinet ist ein verteilter Netzwerkmanagement-Dienst, der es Netzwerkanwendungen erlaubt, Gruppen von Netzwerkteilnehmern (Teilnetze) anzufordern und diese – sofern sie abgeschaltet sind – aufzuwecken. In diesem Kapitel werden Konzept und Funktionsweise im Detail erläutert. In Kapitel 9 und 10 wird eine Implementierung von *Multinet* beschrieben und die Funktionsfähigkeit mittels des Simulationsframeworks OMNeT++ validiert.

Multinet ist ein Software-Dienst und -Protokoll, implementiert auf jedem Knoten im Netzwerk. Softwarekomponenten können über die Dienstschnittstelle Teilnetze anhand einer eindeutigen Kennung anfordern und wieder freigeben. Unter Teilnetz oder Teilnetzwerk wird im weiteren Verlauf eine Gruppe von Knoten (mindestens einer) im gesamten Netzwerk verstanden. Knoten können Teil mehrerer Teilnetze sein, wodurch unterschiedliche Partitionierungsschemata möglich werden. Auf die Unterschiedlichen Aspekte bei der Partitionierung wird in Abschnitt 8.9 eingegangen. Angeforderte Teilnetze werden durch *Multinet* mittels Netzwerkpaketen wachgehalten und schlafende Knoten nachbarweise aufgeweckt. Knoten, die sich in keinem gerade angeforderten Teilnetz befinden, können sich nach eigenem Ermessen abschalten (Zustandsmanagement des jeweiligen Knotens). Der Anforderungszustand aller Teilnetze wird über das gesamte Netzwerk synchronisiert und lokal auf den Knotenzustand der jeweiligen Nachbarn angewendet. Das Konzept hinter *Multinet* wurde als Patent angemeldet [100], auf Fachtagungen vorgestellt ([89], [92]) und in [93] veröffentlicht. Die grundlegenden Ideen flossen im durch das Bundesministerium für Bildung und Forschung geförderte Projekt SEIS (Sicherheit in Eingebetteten, IP-Basierten Systemen) ein ([97, 91]).

Multinet umfasst zwei Kernaspekte:

- Das Wecken von Nachbarn: Wird ein Teilnetz angefordert, weckt jeder Knoten die zum Teilnetz zugehörigen Nachbarn.
- Das Wachhalten eines Teilnetzes: Ein Knoten, der ein Teilnetz wachhält, sendet periodisch Wachhalte-Botschaften (*Keepalive*) ins Netz, welche die eindeutige Kennung des Teilnetzes enthalten. Dadurch wissen zum einen alle Knoten des Teilnetzes, dass sie sich nicht ausschalten dürfen. Zum anderen erhält jeder Knoten ein aktuelles Bild über den Anforderungszustand aller Teilnetze und weiß, welche direkten Nachbarn er aufwecken muss.

8.2 Multinet im Netzwerk

Wie auch beim zentral gesteuerten Teilnetzbetrieb-Konzept (vgl. Kapitel 7) wird im Folgenden von einer schleifenfreien Punkt-zu-Punkt-Topologie ausgegangen, wie sie bei Vollduplex-Ethernet verwendet wird. Die grundlegenden Prinzipien sind jedoch auch auf andere Formen wie Bustopologie oder Ringtopologie anwendbar, entsprechende Mechanismen zum physikalischen Wecken von Nachbarn vorausgesetzt. Eine Beispieltopologie ist in Abbildung 8.1 dargestellt. Jeder Knoten ist über eine oder mehrere

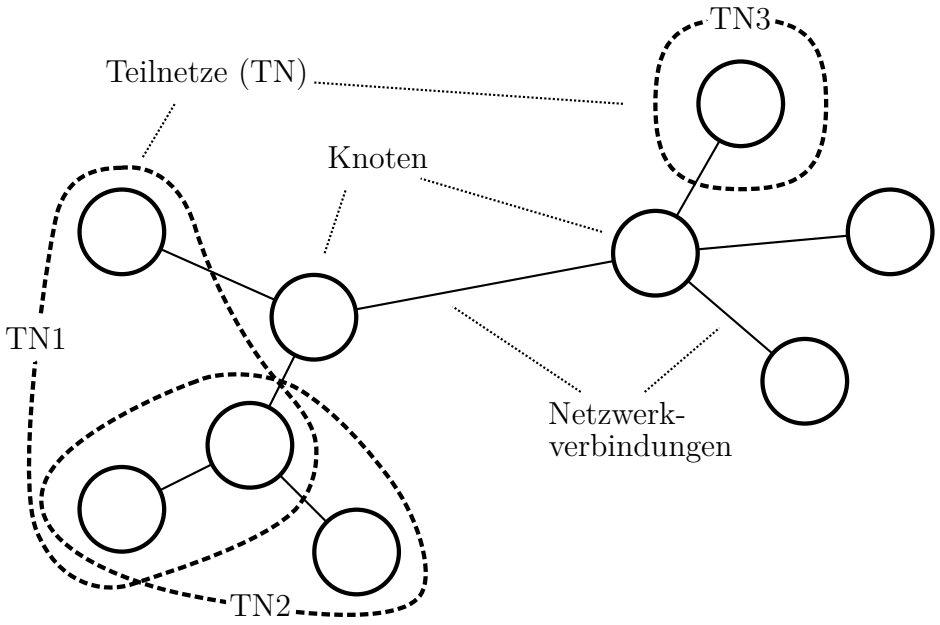


Abbildung 8.1: *Multinet*-Beispieltopologie

Netzwerkverbindungen mit anderen Knoten verbunden. Diese zunächst abstrakte Topologie lässt sich beispielsweise durch ein Ethernet-Netzwerk abbilden, welches aus Endknoten und Switches (*Managed Switches* oder Eingebettete Systeme mit integriertem Ethernet-Switch) besteht. *Multinet* ist als Netzwerkmanagement-Dienst auf jedem Knoten implementiert, und Software-Komponenten (z. B. Anwendungsprozesse) können über die Dienstschnittstelle Teilnetze anfordern und freigeben. Über das Betriebssystem oder das Zustandsmanagement des Knotens kann *Multinet* das Abschalten des Knotens erlauben oder unterbinden (z. B. indem es beim Zustandsmanagement den entsprechenden Betriebszustand anfordert).

Dafür lassen sich folgende zwei Bedingungen aufstellen:

- Einschaltbedingung: Ein Knoten *muss* eingeschaltet bleiben, solange mindestens eines der Teilnetze angefordert wird, denen er zugehörig ist oder in dessen Kommunikationspfad er liegt.
- Ausschaltbedingung: Ein Knoten *darf* sich abschalten, wenn keines der Teilnetze angefordert wird, denen er zugehörig ist, oder in dessen Kommunikationspfad er liegt.

Multinet zeigt bei erfüllter Ausschaltbedingung an, dass der Knoten aus Sicht des Netzwerkmanagements nicht mehr benötigt wird und erlaubt ein Abschalten. Wird der Knoten lokal durch andere Softwarekomponenten oder das Betriebssystem wachgehalten, wird dies durch *Multinet* nicht gestört. Die Aggregation aller möglicher Anforderungen an den Betriebszustand des Knotens erfolgt beim Zustandsmanagement. Am Beispiel AUTOSAR wäre dies wie in Kapitel 4.3 beschrieben der *ECU State Manager* (EcuM), bei dem *Multinet* bei erfüllter Einschaltbedingung den Betriebszustand RUN anfordern würde. Damit fügt sich *Multinet* in das Modell der Steuerung des Betriebszustandes eingebetteter Geräte unproblematisch ein. Eine mögliche Integration in AUTOSAR ist in Kapitel 8.11 genauer beschrieben.

8.3 Zustandskoordination und nachbarweises Wecken

Die netzwerkweite Koordination des Anforderungszustandes von Teilnetzen erfolgt nach dem Prinzip periodischer Wachhaltebotschaften (vgl. Abschnitt 8.4). Sie werden von Knoten versendet, die ein Teilnetz anfordern und alle aktiven Knoten können darüber mitverfolgen, welche Teilnetze zum aktuellen Zeitpunkt wachgehalten werden.

Aus dieser Information wird der gewünschte Betriebszustand aller eigenen, direkten Nachbarn jedes Knotens abgeleitet, d.h. entsprechende, ggf. abgeschaltete Nachbarn werden aufgeweckt. Dazu besitzt jeder Knoten notwendiges (nicht vollständiges) Wissen über Topologie und Partitionierung, also darüber, welcher der eigenen Nachbarn wach sein muss, wenn ein bestimmtes Teilnetz angefordert wird. Für jedes bekannte Teilnetz wird deswegen eine Zuordnung zu einer oder mehreren Netzwerkschnittstellen (bzw. Nachbarn) hinterlegt, ganz ähnlich einer Routingtabelle. Dabei muss der unmittelbare Nachbar nicht selbst Element des jeweiligen Teilnetzes sein. Auch wenn er auf dem Pfad zu diesem Teilnetz liegt, muss er geweckt werden (vgl. Abschnitt 8.6.1). Abbildung 8.2 zeigt beispielhaft ein Netzwerk mit fünf logischen Teilnetzen und die entsprechenden Teilnetztabellen der Knoten D, E und H. Die Netzwerkschnittstellen jedes Knotens sind in der Darstellung jeweils fortlaufend als α, β, γ, usw. gekennzeichnet. Größere Teilnetze mit mehreren kaskadierten Knoten werden schrittweise, *Hop-by-Hop* aufgeweckt, da jeder schlafende Knoten zunächst selbst aufwachen muss, bevor er seine Nachbarn wecken kann. Dies wird in Abschnitt 8.5 detailliert beschrieben. Für die Einteilung eines Netzwerks in mehrere Teilnetze gibt es verschiedene Schemata und Gesichtspunkte, auf die in Abschnitt 8.9 eingegangen wird. Da jeder Knoten in mehreren Teilnetzen sein kann, ist es möglich, unterschiedliche Partitionierungsschemata gleichzeitig zu realisieren.

8.4 Wachhalten von Teilnetzen

Das Wachhalten von Knoten eines Teilnetzes wird durch zyklische Netzwerkpakete realisiert, im Folgenden *Keepalive*-Botschaften genannt.

8.4.1 Zyklische Keepalive-Botschaft

Der Zustand eines Teilnetzes gilt als aktiv solange der Knoten, der es wachhalten möchte, periodisch *Keepalive*-Pakete verschickt. Die Pakete werden als Broadcast an das gesamte Netz verschickt, so dass jeder Knoten sie empfangen kann und den Zustand der Teilnetze entsprechend kennt. In der Botschaft ist neben der Kennung des Teilnetzes auch eine Priorität enthalten, die zur Arbitrierung (vgl. Abschnitt 8.4.2) verwendet wird. Der Paketaufbau der *Keepalive*-Nachricht ist in Anhang B.2 dargestellt. Der Grund für periodisches, aktives Wachhalten im Gegensatz zu einem Benachrichtigungspaket bei Freigabe des Netzes ist, dass diese Benachrichtigung unter

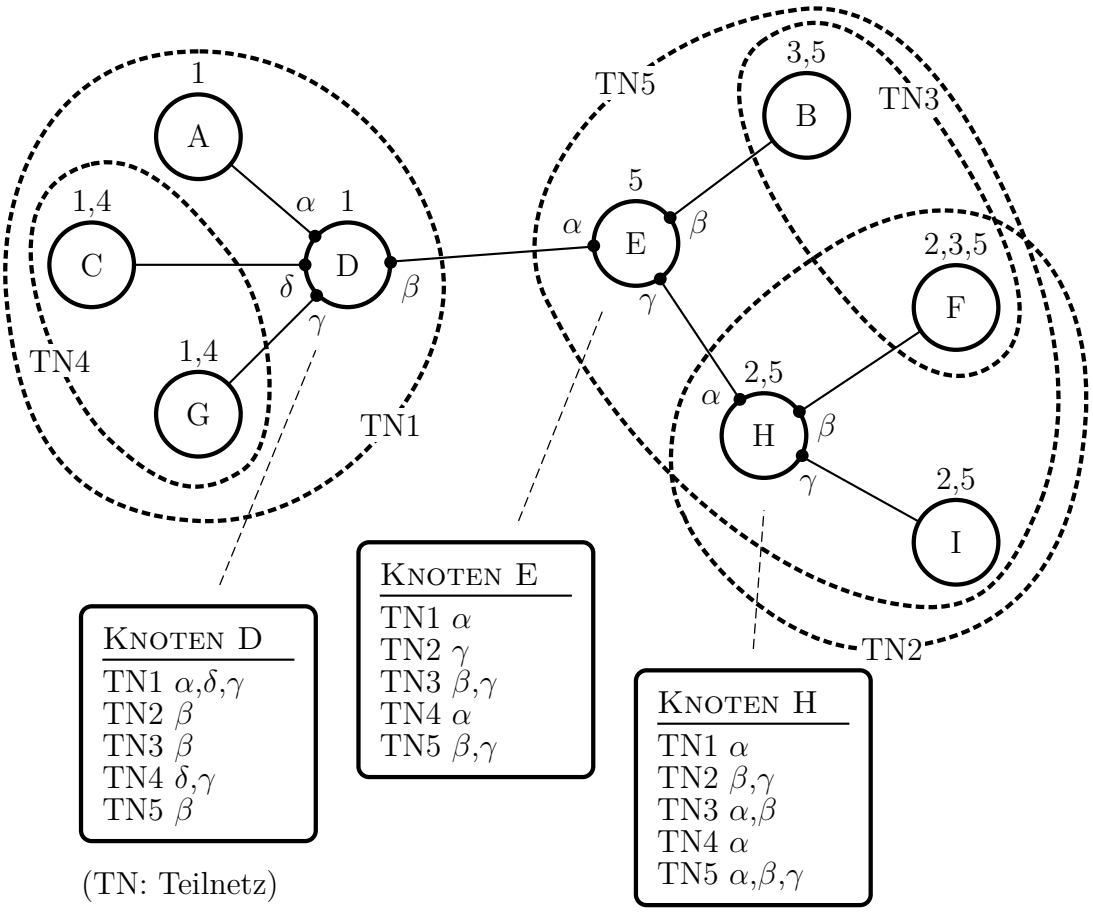


Abbildung 8.2: Beispieltopologie mit den Teilnetztabellen der Knoten D, E und H

Umständen verloren gehen kann und nicht alle Mitglieder erreicht. Dies würde zu einem inkonsistenten Zustand des Teilnetzes im Netzwerk führen und im schlimmsten Falle dazu, dass einzelne Knoten nach Freigabe des Teilnetzes nicht schlafen gehen. Ein solches Verhalten darf beim Einsatz im Fahrzeug nicht auftreten, da Steuergeräte, die fälschlicherweise wach blieben, die Batterie im Stand leeren würden. Dieses Problem ist auch durch eine mögliche Quittierung nicht einfach zu lösen, da dies die Kenntnis über alle Knoten und zu erwartenden Quittierungen im Teilnetz voraussetzen würde. Aus diesen Gründen werden auch beim heute im Fahrzeug eingesetzten Netzwerkmanagement (AUTOSAR, vgl. Kapitel 4.3) Steuergeräte aktiv durch den Bus wachgehalten und legen sich bei eingestellter Buskommunikation selbstständig schlafen.

Jeder Knoten verwaltet für alle ihm bekannten Teilnetze einen Timer, der durch empfangene *Keepalive*-Botschaften rückgesetzt wird. Werden für eine gewisse Zeit keine *Keepalive*-Botschaften mehr für ein Teilnetz empfangen, wird der Zustand des Teilnet-

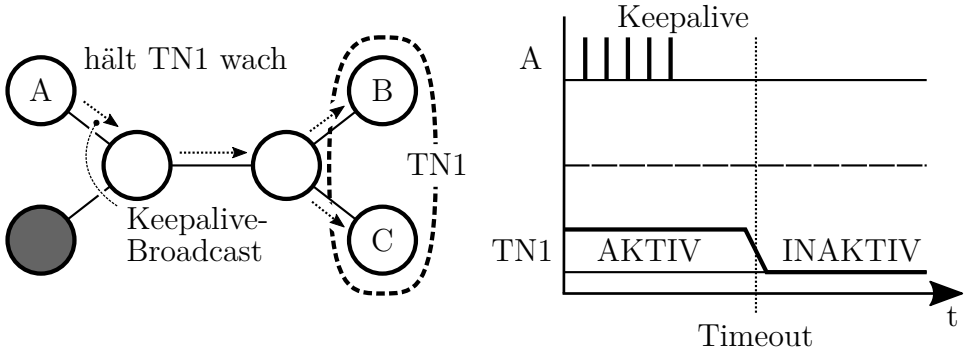


Abbildung 8.3: Wachhalten einer Gruppe durch periodische *Keepalive*-Botschaften

zes inaktiv. Diese Zeit wird im Folgenden *Time-to-live* (TTL) bezeichnet. Auf die Dimensionierung der TTL in Abhängigkeit von der Zykluszeit der *Keepalive*-Botschaften wird in Abschnitt 8.4.3 eingegangen. Abbildung 8.3 veranschaulicht das Wachhalten eines Teilnetzes. Im dargestellten Beispiel hält Knoten A das Teilnetz 1, bestehend aus Knoten B und C wach, indem er *Keepalive*-Botschaften sendet.

Da *Keepalive*-Pakete als Broadcast für alle Knoten sichtbar sind, ergibt sich so ein konsistenter Anforderungszustand aller Teilnetze für alle Knoten des Netzwerks, auch bei sporadischen Paketverlusten.

8.4.2 Arbitrierungsverfahren

Wird ein Teilnetz durch mehrere Knoten gleichzeitig angefordert, so gibt es hinsichtlich des Wachhaltens zwei Möglichkeiten. Entweder senden alle anfordernden Knoten *Keepalive*-Pakete für das selbe Teilnetz, oder es wird ein Arbitrierungsverfahren eingesetzt, so dass nur jeweils einer der Knoten das Wachhalten aktiv übernimmt. Der Vorteil einer solchen Arbitrierung ist eine Entlastung des Netzwerks hinsichtlich der durch *Multinet* verursachten Netzwerk- und Interruptlast. Diese wird in Abschnitt 8.4.4 genauer betrachtet. Für jeden Knoten wird eine Prioritätszahl festgelegt, die als Parameter in den *Keepalive*-Botschaften mitgesendet wird. Senden nun zwei oder mehr Knoten *Keepalive* für das selbe Teilnetz, gewinnt der Knoten mit der höherprioren (geringeren) Prioritätszahl und hält fortan das Teilnetz aktiv wach, während die niederprioreren Knoten in einen passiven Wachhaltemodus wechseln und nicht mehr selbst *Keepalive*-Nachrichten senden, sondern nur noch beobachten, ob das Teilnetz noch wachgehalten wird. Gibt der aktiv wachhaltende Knoten das Teilnetz wieder frei, stellt er das Senden der *Keepalive*-Nachrichten ein. Die verbleibenden Knoten erkennen das Ausbleiben der Botschaften innerhalb eines definierten Erkennungszeitraums und beginnen noch vor

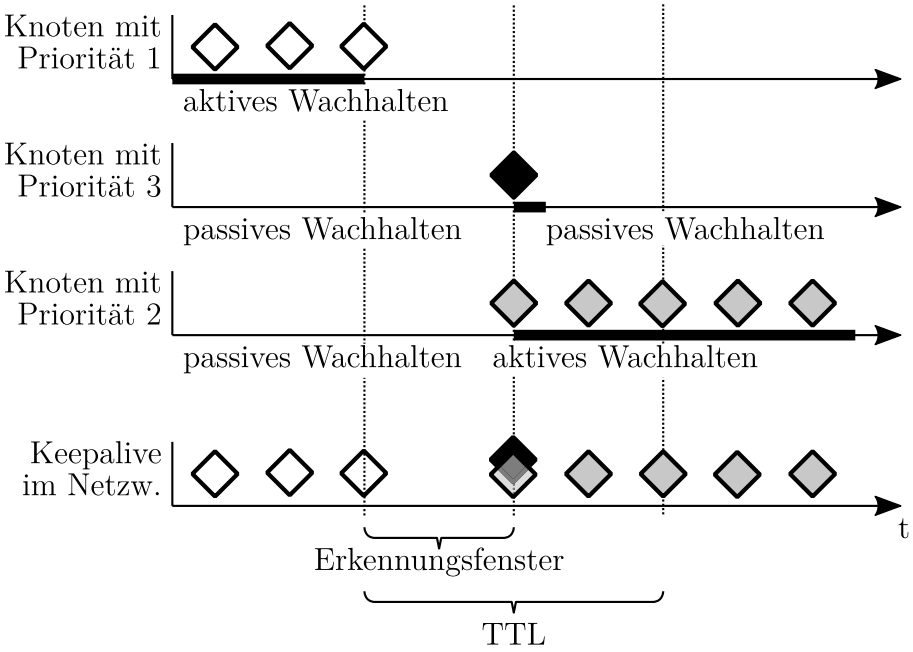


Abbildung 8.4: Arbitrierungsverfahren der *Keepalive*-Botschaft

Ablauf der TTL selbst mit dem Senden von *Keepalive*. Hierdurch ergibt sich wieder eine Arbitrierungsphase, bei der der höherpriore der verbleibenden Knoten übrig bleibt und das Wachhalten des Teilnetzes fortsetzt. Der Arbitrierungsvorgang ist in Abbildung 8.4 veranschaulicht. Durch die passende Wahl des Erkennungszeitraums in Abhängigkeit von der TTL geschieht die Übernahme des Wachhaltens vor Ablauf der TTL, so dass das Teilnetz nahtlos angefordert bleibt.

8.4.3 Zykluszeit, Erkennungsfenster und TTL

Die TTL und das Erkennungsfenster hängen beide von der Zykluszeit der *Keepalive*-Botschaft ab, und müssen groß genug gewählt sein, dass Paketverluste ausgeglichen werden können. So sollte der Verlust einer einzelnen Botschaft nicht gleich den Arbitrierungsprozess starten. Nach Ablauf des Erkennungsfensters sollte auch ein Paketverlust während der Arbitrierungsphase nicht zum Ablaufen der TTL, also zum Timeout des Teilnetzes führen. Gleichzeitig darf das Einschlafen der Knoten auch nicht zu lange dauern, da dies die Effektivität von Teilnetzbetrieb verringert. Dies kann natürlich über kurze *Keepalive*-Zyklen erreicht werden, die Zykluszeit wiederum beeinflusst jedoch die durch *Multinet* verursachte Netzwerk- und Prozessorlast der einzelnen Knoten.

Es gibt also drei gegensätzliche Eigenschaften:

- Responsivität: Einschlafen nach kurzer Zeit
- Geringe Netzwerk- und Prozessorlast: Hohe Zykluszeit der Botschaften
- Hohe Toleranz gegenüber Paketverlusten: TTL mehrere Zyklen lang

Zykluszeit, TTL und Erkennungsfenstergröße sind paramterierbare Werte, die auch einsatzspezifisch angepasst werden können. Um gegenüber Einfachfehlern resistent zu sein, wird für den weiteren Verlauf der Arbeit eine Erkennungsfenstergröße von zwei *Keepalive*-Zykluszeiten gewählt. Weitere zwei Zykluszeiten sollen für die Arbitrierung zur Verfügung stehen, so dass die TTL insgesamt vier Zykluszeiten beträgt. Für die Wahl der Zykluszeit selbst soll im Folgenden zunächst eine Betrachtung der Netzwerklast vorgenommen werden.

8.4.4 Betrachtung der Netzwerklast

Die Netzwerklast, die *Multinet* verursacht, hängt von zwei Faktoren ab: Von der Anzahl an gerade angeforderten Teilnetzen k und von der Zykluszeit der *Keepalive*-Botschaften t_{KA}. Für jedes angeforderte Teilnetz sendet ein Knoten zyklisch *Keepalive*-Botschaften in das Netzwerk. Pro Teilnetz resultiert daraus eine konstante Netzwerklast R_{KA}, die sich aus der Zykluszeit t_{KA} und der Größe der *Keepalive*-Botschaft N berechnet.

$$R_{KA} = \frac{N}{t_{KA}} \tag{8.1}$$

Die Paketgröße N ist abhängig von der Netzwerktechnologie. Sie entspricht bei Ethernet der minimalen Paketgröße von 64 Byte inkl. Paketkopf und zusätzlichen 12 Byte *Interframe Gap*, insgesamt also 76 Byte. Die Botschaft beinhaltet nur wenige Bytes an Daten (vgl. Definition in Anhang B.2), ist damit kleiner als die minimale Nutzdatenlänge 46 Byte und wird auf diese aufgefüllt. Damit lässt sich der dritte Parameter festlegen, unter Berücksichtigung der dadurch verursachten Netzwerklast. Für die in Kapitel 9 und 10 besprochene Implementierung wird eine Zykluszeit von 500 ms verwendet. Damit erfolgt der Timeout von Teilnetzen nach vier *Keepalive*-Zyklen innerhalb von zwei Sekunden, bei einer Netzwerklast von 1 kbit/s pro Teilnetz. Bei einer Partitionierung, bei der jedes Steuergerät ein eigenes Teilnetz darstellt, beträgt bei einer typischen Architektur mit ca. 100 Knoten die Netzwerklast im schlimmsten Fall (jedes Teilnetz

wird gleichzeitig einzeln angefordert) 100 kbit/s. Bei 100BASE-TX Ethernet macht dies ein Promille der verfügbaren Datenrate aus.

8.5 Aufwecken von Teilnetzen

Bereits ins Abschnitt 8.3 wurde beschrieben, dass das Wecken von Teilnetzen nachbarweise – *Hop-by-hop* –, auf Basis der Teilnetztabelle erfolgt. Wird ein Teilnetz angefordert, erhalten alle aktiven Knoten im Netzwerk diese Information durch die *Keepalive*-Botschaft und können sofort entsprechende Nachbarn (laut Teilnetztabelle) wecken. So geweckte Knoten erhalten die Teilnetzanforderung über die periodischen *Keepalive*-Botschaften und wecken ihrerseits weitere Nachbarn auf, wodurch sich das Aufwachen des Teilnetzes nachbarweise fortsetzt. Abbildung 8.5 veranschaulicht den Vorgang anhand eines Beispielnetzwerks, das durch den Knoten an der Wurzel des Baumes angefordert wird. Jeder aufgewachte Knoten kann erst Nachbarn wecken,

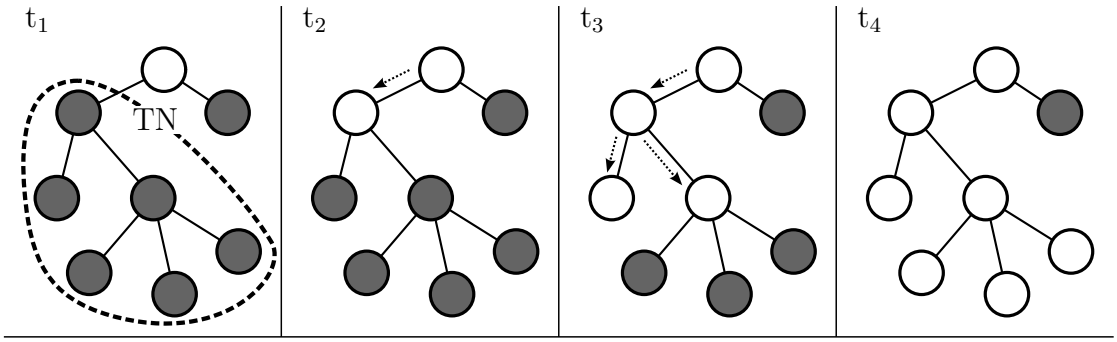


Abbildung 8.5: Schrittweises Wecken eines Teilnetzes

wenn er weiß, welche Teilnetze angefordert werden. Müsste er auf die nächste, reguläre *Keepalive*-Botschaft warten, würde dies das Aufwachen des Teilnetzes bei jedem *Hop* um schlimmstenfalls eine *Keepalive*-Zykluszeit verzögern – im hier gewählten Falle also um bis zu 500 ms. Deswegen sendet jeder Knoten unmittelbar nach dem Aufwachen eine Botschaft ins Netzwerk (im Folgenden *Query* genannt), mit der er alle anderen zum Verschicken einer sofortigen, asynchronen *Keepalive*-Nachricht auffordert. Dadurch muss der gerade aufgewachte Knoten nicht erst die *Keepalive*-Zykluszeit abwarten, sondern erfährt nahezu sofort, welche Teilnetze gerade angefordert sind und kann entsprechende Nachbarn aufwecken. Abbildung 8.6 verdeutlicht das Zeitverhalten beim Aufwecken eine Teilnetzes jeweils ohne und mit *Query* und asynchroner *Keepalive*-Botschaft. Die Verzögerungszeit beim Wecken von Teilnetzen hängt dann nur noch von

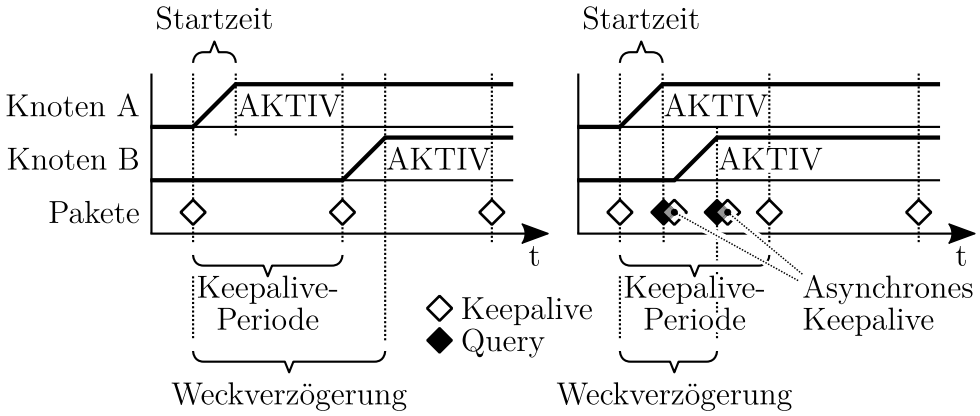


Abbildung 8.6: Weckzeit ohne (links) und mit (rechts) asynchroner *Keepalive*-Botschaft

der Netzwerktopologie ab. Bei der Auslegung eines Netzwerks muss diese Verzögerung berücksichtigt werden. Lange, kaskadierte Ketten von Knoten sind unter diesem Gesichtspunkt unvorteilhaft. Sterntopologien sind besser, da die meisten Knoten parallel aufwachen können. Insbesondere bei Anwendungen, die eine sehr kurze Reaktionszeit erfordern, muss die Anzahl der Knoten im Pfad beim Systementwurf berücksichtigt werden, und ggf. eine Topologie gewählt werden, bei der die geforderte Reaktionszeit eingehalten wird. Oftmals wird die Kerninfrastruktur von Netzwerken (Backbone) in den meisten Situationen ohnehin benötigt, so dass diese Knoten nicht oder nur selten schlafen. Damit beschränkt sich für viele anwendungsorientierte Topologien die Kaskadierungstiefe auf eine oder zwei Ebenen, wodurch sich auch die Weckverzögerung in Grenzen hält. Die *Keepalive*-Botschaft erreicht über den Backbone alle Teile des Netzwerks, Endknoten wachen sofort auf und untergeordnete Teilnetze binnen weniger *Hops*. Dies ist veranschaulichend in Abbildung 8.7 dargestellt.

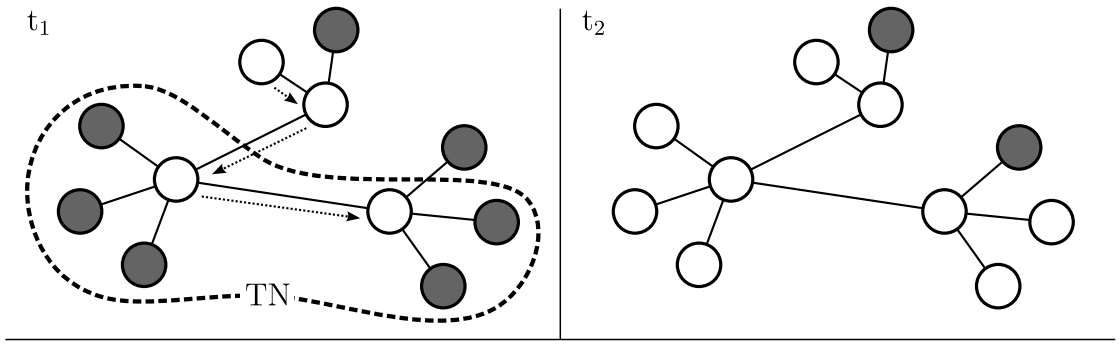


Abbildung 8.7: Sofortiges Aufwachen angeforderter Teilnetze bei aktivem Backbone

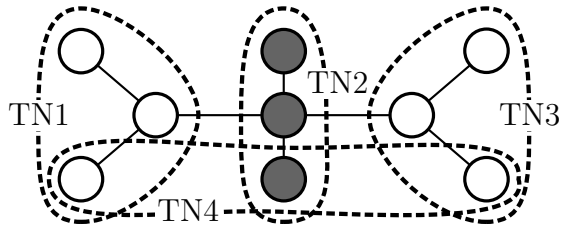


Abbildung 8.8: Separation des Netzwerks durch schlafenden Knoten

8.6 Einschlafen von Teilnetzen

Läuft die TTL eines Teilnetzes ab, dürfen sich die Knoten des Teilnetzes schlafen legen – vorausgesetzt, sie befinden sich nicht noch in anderen, angeforderten Teilnetzen. Dieser Zeitpunkt kann auf verschiedenen Knoten aufgrund nicht synchroner Uhren leicht variieren. Um dies auszugleichen, muss ein Knoten mit dem Abbauen der Netzwerkverbindungen und dem Abschalten noch mindestens eine Zykluszeit lang warten, da ansonsten ein Nachbar, bei dem die TTL ggf. noch nicht abgelaufen ist, den Verlust der Verbindung als Fehler interpretiert.

8.6.1 Problem der Netzseparation

Das Abschalten von nicht angeforderten Teilnetzen, bzw. Knoten, kann jedoch zu einer Teilung des Netzwerkes in mehrere, voneinander losgelöste Inseln führen. Die Problematik wird in Abbildung 8.8 ersichtlich. Der Knoten in der Mitte von Teilnetz 2 verbindet die wachen Teilnetze 1 und 3 miteinander. Schaltet er sich aus (weil Teilnetz 2 nicht angefordert wird), entstehen zwei voneinander losgelöste Netze, die nicht mehr miteinander kommunizieren können. Ebenso gäbe es keinen zusammenhängenden Pfad zwischen den Knoten des Teilnetzes 4. Eine solche Separation des Netzwerks darf nicht

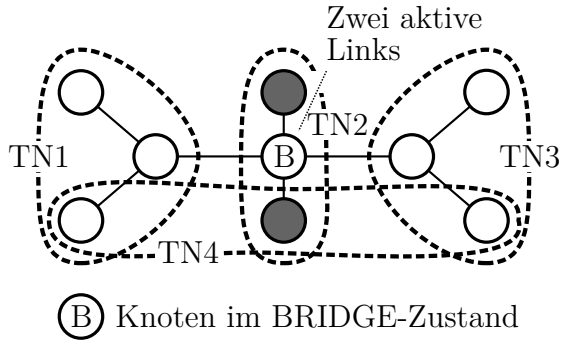


Abbildung 8.9: Wachbleiben des nicht angeforderten Knotens im BRIDGE-Zustand

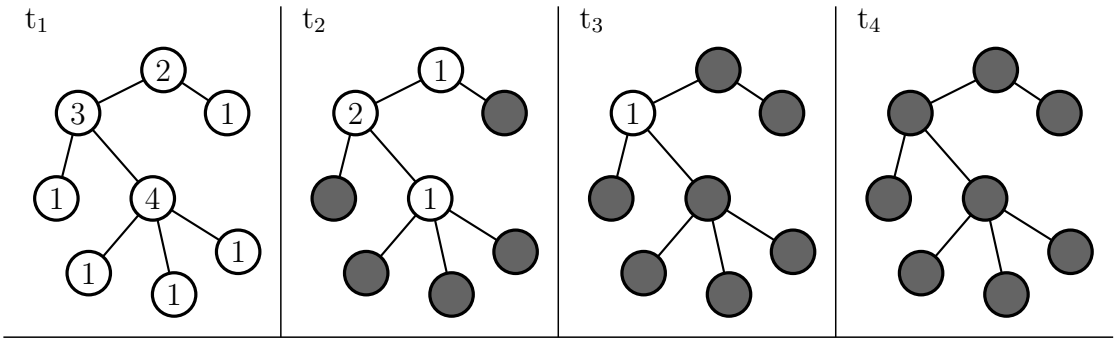


Abbildung 8.10: Einschlafen eines Teilnetzes von außen nach innen

auftreten. Deswegen dürfen sich Knoten nicht abschalten, wenn sie mehrere aktive Netzbereiche koppeln, selbst wenn sie sich selbst in keinem angeforderten Teilnetz befinden. Konkret bedeutet das, dass sich Knoten erst dann ausschalten dürfen, wenn nur noch eine ihrer Netzwerkverbindungen aktiv ist, sie also ein Endknoten sind. Dies wird durch den Zustand BRIDGE im Zustandsdiagramm (vgl. Kapitel 8.7) realisiert. Im vorher beschriebenen Fall bleibt der Knoten in der Mitte also wach und brückt die beiden aktiven Teilnetze 1 und 3 (vgl. Abbildung 8.9). Im Ergebnis bedeutet das, dass sich das Netzwerk stets von seinen Endknoten aus nach innen abbaut. Verzweigungen (Switches) werden zu Endknoten, sobald jeder ihrer Äste abgeschaltet ist, und schalten sich danach wiederum selbst aus, und so fort. Abbildung 8.10 zeigt dies anhand eines Beispiels. Neben den Knoten ist jeweils die Anzahl der noch aktiven Netzwerkverbindungen vermerkt. Auf diese Weise ist sichergestellt, dass es zu keinem Zeitpunkt zu einer Netzseparation durch Abschalten kommt. Werden zwei äußere Knoten unabhängig voneinander aufgeweckt, wächst das Netzwerk von beiden Seiten aus zusammen, sobald ein Knoten ein Teilnetz auf der 'anderen' Seite anfordert oder ein Teilnetz beide Seiten umspannt (vgl. Abbildung 8.11). Knoten, die zwischen den beiden Teilnetzen

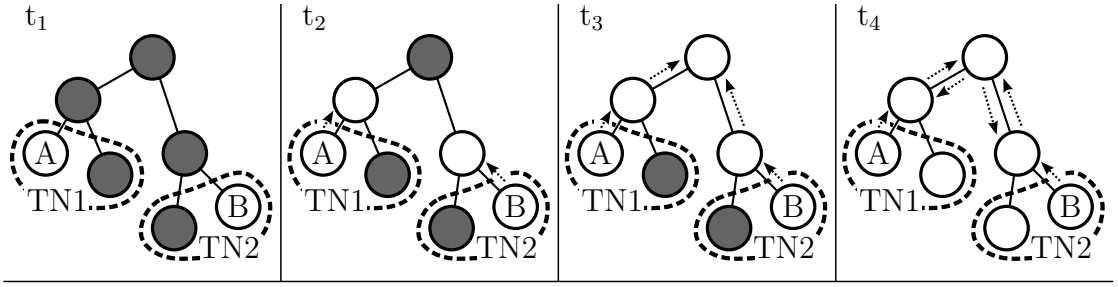


Abbildung 8.11: Zusammenwachsen zweier unabhängig aufgeweckter Netzbereiche

liegen, werden aufgeweckt und bleiben auch hier im BRIDGE-Zustand wach, auch wenn sie sich selbst in keinem angeforderten Teilnetz befinden.

8.6.2 Offline-Betrieb von Knoten

Aus Sicht des Netzwerkmanagements bedeutet schlafen legen nicht zwangsweise das sofortige Abschalten des Knotens, sondern zunächst das Herauslösen aus dem Netzwerk. In diesem Offline-Zustand ist der Knoten aus Netzwerksicht abgeschaltet, kann aber tatsächlich durch andere Prozesse noch physisch wachgehalten werden. *Multinet* löst also den Knoten aus dem Netzwerk, sobald keine Kommunikation (kein Teilnetz) mehr angefordert wird, indem es die Netzwerkverbindungen koordiniert abbaut, und hört auf, den Knoten beim Zustandsmanagement wach zu halten. Das tatsächliche Abschalten des Knotens, nachdem dieser aus dem Netzwerk herausgelöst wurde, erfolgt durch dessen Zustandsmanagement, nachdem *Multinet* die Anforderung des Betriebszustandes zurücknimmt. Das Netzwerkmanagement ist aber nur einer von mehreren möglichen Nutzern des Zustandsmanagements. Auch andere Softwarekomponenten können ggf. einen aktiven Betriebszustand des Knotens anfordern. *Multinet* verbleibt dann in einem passiven Zustand, bis entweder wieder Kommunikation angefordert wird, oder sich der Knoten abschaltet.

8.7 Zustandsdiagramm

Die zahlreichen Systemzustände von *Multinet* können als endlicher Automat modelliert und gesteuert werden. Abbildung 8.12 zeigt das Zustandsdiagramm. Die möglichen Zustände sind im Folgenden kurz beschrieben.

Der Zustand OFFLINE ist der Grundzustand nach dem Aufwachen des Knotens oder nach dem Abbauen aller Netzwerkverbindungen und bleibt bestehen, solange kein Netzwerkmanagement vonnöten ist, also solange keine Netzwerkkommunikation angefordert wird. Wurde der Knoten durch einen Nachbarn geweckt, bzw. versucht ein Nachbar, eine Netzwerkverbindung aufzubauen wechselt der Zustand zu REMOTE WAKE. Wird lokal die Kommunikation mit einem Teilnetz durch eine Softwarekomponente angefordert[1], wechselt der Zustand direkt zu NETWORK REQUESTED.

[1] Anmerkung: Wacht beispielsweise ein Knoten durch ein lokales Ereignis (Tastendruck, Funksignal, ...) auf, bleibt der Knoten im Zustand OFFLINE bis eine lokale Softwarekomponente Netzwerkkommunikation anfordert.

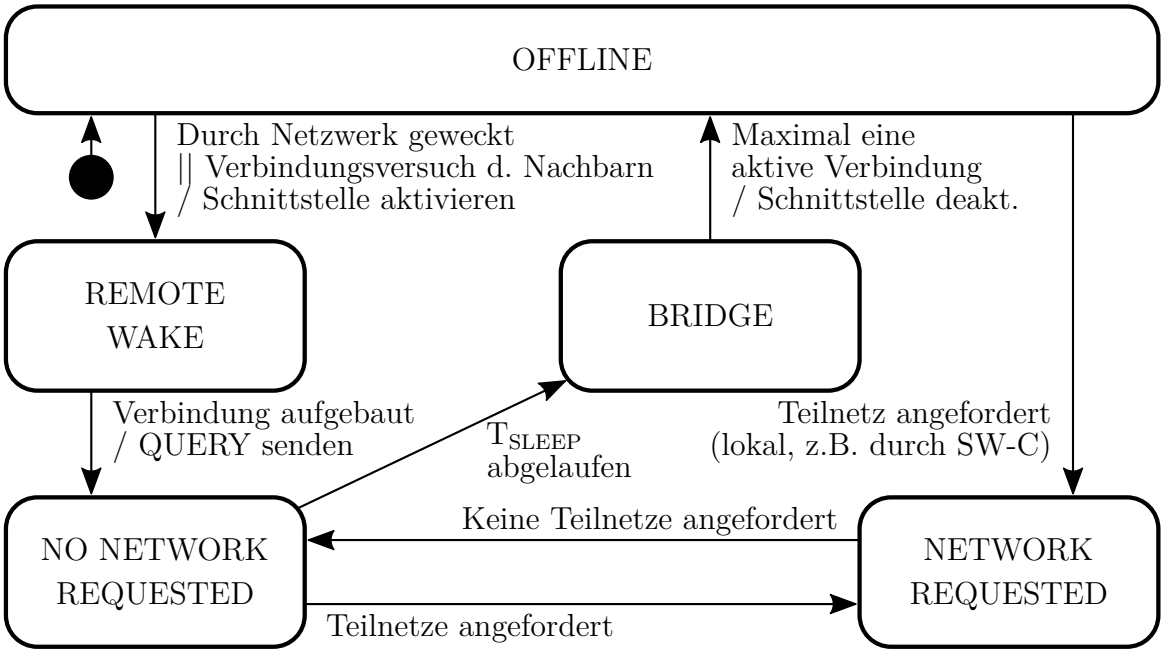


Abbildung 8.12: Zustandsdiagramm von *Multinet*

Der Zustand REMOTE WAKE wird eingenommen, wenn der Knoten durch einen seiner Nachbarn aufgeweckt wurde oder wenn ein Verbindungsversuch von außen erkannt wird. Beim Wechsel in den Zustand REMOTE WAKE wird die entsprechende Netzwerkschnittstelle aktiviert und auf den erfolgreichen Verbindungsaufbau zu dem Nachbarn gewartet. Sobald die Verbindung hergestellt wurde, wird das *Query*-Paket im Netzwerk versendet und in den Folgezustand NO NETWORK REQUESTED gewechselt.

Im Zustand NO NETWORK REQUESTED befindet sich der Knoten, wenn kein Teilnetz, dem er zugehörig ist, angefordert wird, also wenn die TTL jener Teilnetze Null ist. Beim Eintreten in diesen Zustand wird der Timer T_SLEEP gestartet, nach dessen Ablaufen der Knoten sich aus dem Netzwerk abmeldet und seine Netzwerkverbindungen herunterfährt, oder nur noch als Vermittlerknoten im Netzwerk wach bleibt (im Zustand BRIDGE). In jedem Fall muss T_SLEEP mindestens eine *Keepalive*-Zykluszeit lang sein, um das asynchrone Ablaufen der TTL auf verschiedenen Knoten auszugleichen, wie bereits in Abschnitt 8.6 erläutert wurde. Wird im Zustand NO NETWORK REQUESTED ein Teilnetz angefordert (beispielsweise lokal durch eine Softwarekomponente oder durch den Empfang einer *Keepalive*-Nachricht), wechselt der Zustand zu NETWORK REQUESTED.

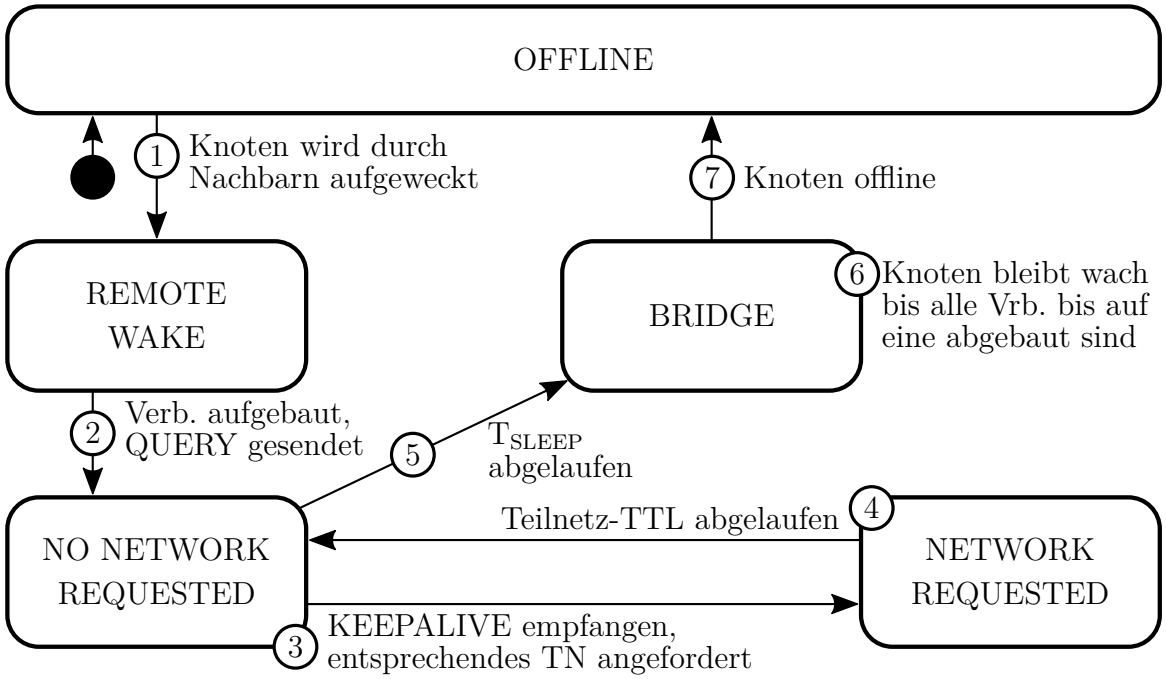


Abbildung 8.13: Weg durch das Zustandsdiagramm bei Anforderung eines Teilnetzes durch einen anderen Knoten

Der Zustand NETWORK REQUESTED stellt den normalen Betriebszustand eines Knotens dar. *Multinet* bleibt in diesem Zustand, solange mindestens eines der Teilnetze, denen der Knoten angehört, angefordert wird, sei es lokal oder über das Netzwerk. Die TTL aller Teilnetze zählt herunter und wird jeweils durch eintreffende *Keepalive*-Pakete zurückgesetzt. Wird keines dieser Teilnetze mehr angefordert (die TTL dieser Teilnetze wird Null), wechselt der Zustand wieder in NO NETWORK REQUESTED und der Timer T_SLEEP wird gestartet.

Läuft der Timer T_SLEEP im Zustand NO NETWORK REQUESTED ab, ohne dass Teilnetze angefordert werden, wechselt der Knotenzustand zu BRIDGE. Im Zustand BRIDGE bleibt ein Knoten solange er mehr als eine aktive Netzwerkverbindung hat, also mehrere aktive Netzsegmente brückt. Dies dient der Vermeidung einer Separation des Netzes durch schlafende Knoten (vgl. Abschnitt 8.6.1). Hat der Knoten nur noch eine aktive Netzwerkverbindung, deaktiviert er sie, und wechselt in den Zustand OFFLINE[2]. Abbildung 8.13 zeigt den Ablauf für den Fall, dass ein Knoten geweckt wird, das entsprechende Teilnetz eine Zeit lang angefordert wird, danach freigegeben wird und der Knoten wieder schlafen geht.

[2]Anmerkung: Durch diesen Zustand ist das später beschriebene, von den Endknoten aus beginnende Einschlafen von Teilnetzen realisiert, bei dem Verzweigungsknoten zunächst auf das Einschlafen ihrer Äste warten.

8.8 Verhalten in Fehlerfällen

Bei Fehlern wie Verbindungsabbrüchen, Paketverlusten oder falschem Verhalten eines Knotens sollte der Zustand aller Teilnetze über das Netzwerk dennoch konsistent bleiben und das Verhalten vorhersagbar. Im Folgenden werden mögliche Einfachfehler und das daraus resultierende Systemverhalten diskutiert.

8.8.1 Verlust einer zyklischen Botschaft

Der Verlust einer zyklischen *Keepalive*-Botschaft darf nicht zum irrtümlichen Ablaufen der TTL des Teilnetzes und damit zur Abschaltung von Knoten führen. Wie in Abschnitt 8.4.3 dargestellt, kann dies durch geeignete Wahl der Größe von Erkennungsfenster und TTL sichergestellt werden. Wenn beide Fenster jeweils der doppelten Zykluszeit entsprechen, kann ein einzelner Paketverlust noch keine Arbitrierungsphase auslösen. Auch ein Paketverlust innerhalb einer gerade stattfindenden Arbitrierungsphase führt noch zu keinem Gruppen-Timeout. Durch weitere Erhöhung der Fenstergrößen lassen sich auch Mehrfachfehler ausgleichen, jedoch erhöht dies, wie diskutiert, entweder die durch *Multinet* verursachte Netzwerklast, oder die TTL und damit die Zeit, nach welcher Knoten auch tatsächlich schlafen gehen.

8.8.2 Verlust einer asynchronen Botschaft

Asynchrone *Keepalive*-Botschaften werden in zwei Fällen verschickt, bei Anforderungsbeginn und als Reaktion auf ein *Query*-Paket. Wird ein Teilnetz durch eine Softwarekomponente bei *Multinet* angefordert, so wird zunächst eine asynchrone *Keepalive*-Botschaft versendet, so dass alle wachen Knoten im Netzwerk darüber informiert werden, und entsprechende Nachbarn wecken. Geht diese Nachricht verloren, so erfahren diese erst mit der ersten zyklischen Botschaft von dem neuen Zustand. Die dadurch entstehende Verzögerung beträgt im schlimmsten Fall eine *Keepalive*-Zykluszeit. Dies kann zu unerwünscht hohen Latenzen führen, jedoch nicht zu einem inkonsistenten Zustand im Netzwerk, da der Fehler mit der nächsten Botschaft behoben wird.

Sobald ein Knoten, der über das Netzwerk geweckt wurde, aufgewacht ist, sendet er ein *Query*-Paket (Zustand REMOTE WAKEUP) und fordert damit asynchrone *Keepalive*-Botschaften aller Knoten an, die Teilnetze wachhalten. Geht das *Query* verloren,

entsteht schlimmstenfalls eine Verzögerung von einer *Keepalive*-Zykluszeit, bis der neu hinzugekommene Knoten die angeforderten Teilnetze kennt. Dies trifft auch dann zu, wenn eines der daraufhin verschickten asynchronen *Keepalive*-Pakete nicht ankommt. In keinem Fall kommt es zu einem inkonsistenten Gruppenzustand, da auch hier der Fehler mit der nächsten zyklischen Botschaft behoben wird.

8.8.3 Linkverlust oder Ausfall

Wird ein Knoten durch einen defekten Link vom Netzwerk getrennt, empfängt er keine *Keepalive*-Botschaften mehr. Nach dem daraus resultierenden Timeout darf sich der Knoten ausschalten. Damit ist sichergestellt, dass der jeweilige Knoten nicht irrtümlich wach bleibt. Teilnetze, die dieser Knoten als einziger wachgehalten hat, gehen nach Ablauf deren TTL ebenfalls schlafen. Im Netzwerkmanagement wird der Verbindungszustand zu allen Nachbarn überwacht. Fällt ein Knoten (bzw. die Verbindung zu ihm) aus, kann dies an das Fehlermanagement gemeldet werden. Geeignete Maßnahmen müssen jedoch an anderer Stelle umgesetzt werden, die Fehlerbehebung und Diagnose wird hier nicht als Aufgabe des Netzwerkmanagements verstanden.[3]

8.8.4 Irrtümliches Aufwachen

Wird ein Knoten nicht durch seinen Nachbarn geweckt, sondern durch eine Störung oder Fehlfunktion seines Weckmechanismus, geht dieser nach kurzer Zeit wieder schlafen. Über den Erhalt der *Keepalive*-Botschaften nach seinem *Query*-Paket (Zustand REMOTE WAKEUP), stellt er fest, dass keine relevanten Teilnetze angefordert werden (Zustand NETWORK NOT REQUESTED), und schaltet sich nach Ablaufen des Timers T_SLEEP wieder ab.

8.9 Partitionierung von Teilnetzen

Multinet stellt die Netzwerkmanagement-Infrastruktur zur Implementierung des Konzeptes von Teilnetzen zur Verfügung. Bei Betrachtung eines größeren, komplexen Netz-

[3] Anmerkung: In der Implementierung von *Multinet* ist dennoch eine einfache Diagnose umgesetzt, welche den Linkzustand überwacht und bei unerwarteten Verbindungsabbrüchen einen erneuten Linkaufbau versucht

werks stellt sich dir Frage, nach welchen Gesichtspunkten und auf welche Weise es sich am besten in Teilnetze untergliedern lässt. Dies ist jedoch abhängig vom Einzelfall, es gibt nicht die eine, optimale Partitionierung. Die Möglichkeiten sinnvoller Gruppierungen sind vielfältig: Jeder Knoten könnte ein eigenes Teilnetz darstellen, Teilnetze könnten nach Funktionen gegliedert werden oder auch nach infrastrukturellen Gesichtspunkten (z. B. Backbone). Sinnvoll ist oft eine Kombination mehrerer Partitionierungsschemata, da Knoten (theoretisch) in beliebig vielen Teilnetzen sein können. Auch wenn es keine allgemeingültige Antwort auf die Frage nach der optimalen Partitionierung gibt, sollen im Folgenden dennoch grundlegende Überlegungen angestellt werden. Dabei gibt es zwei grundsätzliche Fragen: Wie stark ist die Verantwortlichkeit für das Wecken und Freigeben von Teilnetzen im Netzwerk verteilt, und wie feingranular ist die Gruppierung einzelner Knoten zu Teilnetzen.

8.9.1 Verteilung der Verantwortlichkeit

Für die Verteilung der Verantwortlichkeit lassen sich drei generelle Formen festhalten: Ein einzelner Knoten kann zentral die Verwaltung aller Teilnetze übernehmen. Ihm obliegt es, Teilnetze je nach Zustand und Situation anzufordern und freizugeben. Der Vorteil hierbei ist eine hohe Vorhersagbarkeit des Systemverhaltens und eine einfache, anwendungsfallbezogene Partitionierung der Knoten auf Teilnetze. Um einen Ausfall des zentralen Knotens abzusichern, kann ein anderer Knoten dessen Funktion im Bedarf übernehmen. Ein einfaches Beispiel im Fahrzeug wäre ein zentrales Steuergerät (z. B. das Fahrzeuggateway), welches die Kommunikation und damit den Systemzustand des Fahrzeugs überwacht und entsprechend Teilnetze anfordert. So könnte beispielsweise ein System zur Einparkhilfe, bestehend aus einem Steuergerät und mehreren Kameras, geweckt werden, sobald die Geschwindigkeit unter einen gewissen Schwellwert (z. B. zehn Stundenkilometer) fällt.[4] Als zweite Möglichkeit kann jeder Knoten Teilnetze vollständig dezentral anfordern und freigeben. Damit geht eine hohe Flexibilität einher, da Teilnetze hier am besten nach einzelnen Funktionen gegliedert und angefordert werden können. Gleichzeitig ist es schwieriger, das Systemverhalten vorherzusagen, da die Verantwortung nicht mehr in einem Punkt konzentriert ist und sich Überlagerungseffekte ergeben. Auch der Aufwand beim Entwurf des Systems ist höher, da für jeden Knoten festgelegt werden muss, wann er welche Teilnetze bzw. Funktionen anfordern muss. Dementsprechend ist auch eine hohe Zahl an Teilnetzen

[4] Die Startzeit eines solchen Systems kann relativ hoch (im Sekundenbereich) sein, so dass durch ein vorzeitiges Wecken sofortige Verfügbarkeit im Bedarfsfall gewährleistet ist.

zu erwarten. Praxisbezogenes Beispiel: Die Blinker im Seitenspiegel werden durch das Türsteuergerät angesteuert, die Blinker in den Front- und Heckleuchten durch die entsprechend platzierten Lichtsteuergeräte. Betätigt der Fahrer den Blinker, fordert das entsprechende Steuergerät (Lenksäule oder Kombiinstrument) das Teilnetz 'Blinkfunktion' an, in dem alle beteiligten Steuergeräte gegliedert sind. Eine dritte Möglichkeit besteht darin, das Netzwerk hierarchisch zu strukturieren. Netzbereiche werden von ausgewählten Knoten zentral verwaltet, d.h. Teilnetze z. B. auf Knoten- oder Subsystemebene, angefordert. Diese ausgewählten Knoten repräsentieren die Teilnetze wiederum in einer größeren Vernetzungsstruktur, die entweder dezentral oder von einem übergeordneten Knoten verwaltet wird. Dieser Mittelweg aus zentraler und dezentraler Verwaltung erlaubt eine praxisnahe Partitionierung des Netzwerks nach Systemen und Subsystemen, die sich gegenseitig steuern, und lässt sich am einfachsten auf die Struktur heutiger Fahrzeugarchitekturen abbilden. Ein Beispiel aus der Fahrzeugtechnik wäre ein Bereich 'Fahrassistenz', der aus Kameras, Radaren und verschiedenen Funktionssteuergeräten besteht, gegliedert in unterschiedliche Teilnetze je nach gewählter Funktion. Das zuständige Zentralsteuergerät fordert diese Funktionen nach Bedarf an, und wird seinerseits beispielsweise durch das Kernnetzwerk geweckt, sobald sich das Fahrzeug bewegt.

Diese drei Möglichkeiten spiegeln direkt die in Kapitel 5 beschriebenen Varianten der Verteilung von Netzwerkmanagement wieder. Im realen Fahrzeugbordnetzwerk ist wahrscheinlich eine Kombination der drei Varianten, je nach Anwendungsfall am zielführendsten. Alle beschriebenen Beispiele sind zwar nur an die Realität angelehnt, könnten jedoch so in einem echten Fahrzeug nebeneinander existieren und durch *Multinet* abgebildet werden.

8.9.2 Granularität

Neben der Verantwortlichkeit für das Netzwerkmanagement gibt es auch für die Granularität der Teilnetze verschiedene Möglichkeiten, und beides geht wie beschrieben oft miteinander einher. Zunächst kann das gesamte Netzwerk als Teilnetz aufgefasst werden. Dann kann jeder Knoten das gesamte Netzwerk aufwecken und wachhalten. Das entspricht der heutigen Umsetzung von CAN-Netzwerkmanagement ohne Teilnetzbetrieb (vgl. Kapitel 4.3). Auf der nächsten Ebene können Teilnetze aufgrund der Topologie und Architektur gebildet werden. So wäre beispielsweise das Kernnetz, welches die Backbone-Infrastruktur zwischen allen Funktionsbereichen im Fahrzeug darstellt,

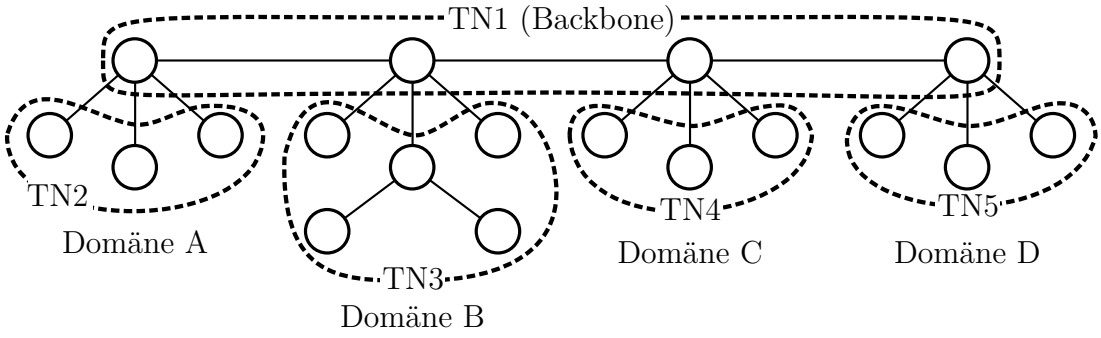


Abbildung 8.14: Partitionierung nach Topologie und Funktionsdomänen

ein eigenes Teilnetz, wie auch jeweils die vom Backbone abzweigenden Äste. Dies spielt besonders gut zusammen mit einer hierarchisch verteilten Verantwortung wie im vorherigen Kapitel beschrieben und ließe sich auf eine vernetzte Domänenarchitektur gut anwenden. Abbildung 8.14 zeigt eine Topologie mit mehreren Funktionsbereichen (Domänen), die durch einen Backbone miteinander vernetzt sind. Der Backbone stellt ein eigenes Teilnetz dar, sowie auch die jeweiligen Domänen. Diese können jeweils selbst nochmals in mehrere Teilnetze partitioniert sein.

Eine funktions- oder serviceorientierte Einteilung gruppiert alle einer jeweiligen Funktion oder einem Service zugehörigen Knoten zu einem Teilnetz. Dies kann gut mit einer serviceorientierten Middleware (z. B. *Service Oriented Middleware over IP*, SOME/IP) kombiniert werden, so dass beim Abonnement eines Service implizit die zugehörigen Knoten geweckt werden. Eine solche Partitionierung stellt die flexibelste, und von der Hardware unabhängigste Form dar, da keine Geräte zu Teilnetzen gruppiert werden, sondern Funktionen. Auf welche realen Knoten die Funktionskomponenten dann verteilt sind, spielt keine Rolle und kann sich von Variante zu Variante unterscheiden – es werden immer die richtigen Knoten geweckt.

Die feinste Granularität stellt eine knotenselektive Partitionierung dar. Ein Knoten ist hier ein eigenes Teilnetz. Auf diese Weise können individuelle Knoten explizit angefordert werden. Bei Systemen mit sehr spezifischen Komponenten und enger Kopplung von Software und Hardware ist diese Variante einfach, übersichtlich und gut vorhersagbar.

Wie auch bei der Verteilung gibt es auch hinsichtlich der Granularität keine einzelne, richtige oder optimale Ausprägung. Vielmehr ist naheliegend, alle Varianten jeweils abgestimmt auf die jeweiligen Anwendungsfälle einzusetzen. Es ist sowohl sinnvoll, ein Teilnetz um den Backbone des Netzes zu spannen, als auch eine serviceorientierte

Gliederung mit der Middleware zu verbinden, oder das eine, spezifische Anhängersteuergerät individuell anzufordern. Da jedes Steuergerät in mehreren Teilnetzen sein kann, können sich diese Partitionierungsschemata auch überlagern.

8.9.3 Aufstellen der Teilnetztabelle

Im allgemeinen Fall kennt jeder Knoten jedes im Netzwerk vorhandene Teilnetz. Bei statischen, geschlossenen Systemen ist der einfachste Weg, die Teilnetztabelle statisch zu konfigurieren, da Topologie und Partitionierung bekannt sind und sich nicht verändern. Mehr Dynamik und Flexibilität bietet ein automatisiertes Lernen der Tabellen. Selbst in einem statischen System vereinfacht das das Variantenmanagement, da nicht für jede Variante eine eigene Version der Tabelle für jedes Gerät erstellt werden muss. Für dynamische Systeme ist ein Erlernen und Überwachen zur Laufzeit unabdingbar. Ein anschauliches Beispiel ergibt sich, wenn man Teilnetze service- oder funktionsorientiert gliedert, aber gleichzeitig Services bzw. Funktionen virtualisiert und im Netzwerk verschiebbar gestaltet. Dann kann sich die Zuordnung von direkten Nachbarn zu Teilnetzen und sogar die eigene Zugehörigkeit eines Knotens zu Teilnetzen verändern, was ein dynamisches Lernen notwendig macht. Das dynamische Erlernen wurde nicht im Rahmen dieser Arbeit umgesetzt und ist Gegenstand weiterer Forschungen.

8.10 Einfluss von Dienstgüte und Auslastung

Eine hohe Auslastung des Netzwerks kann Auswirkungen auf die Weiterleitungsdauer der *Keepalive*-Botschaften und damit zum einen auf das Wachhalten von Teilnetzen und zum anderen auf die Weckverzögerung von Knoten haben. Zur Verdeutlichung des Problems soll ein einfaches Beispiel, wie in Abbildung 8.15 dargestellt herangezogen werden.

8.10.1 Weiterleitung der *Keepalive*-Botschaften

Für die Knoten gelte das folgende, einfache Modell. Jeder Port besitzt eine Eingangs- und eine Ausgangswarteschlange nach dem FIFO-Prinzip (*First In First Out*) für ankommende, beziehungsweise zu versendende Pakete. Switches holen Pakete aus den

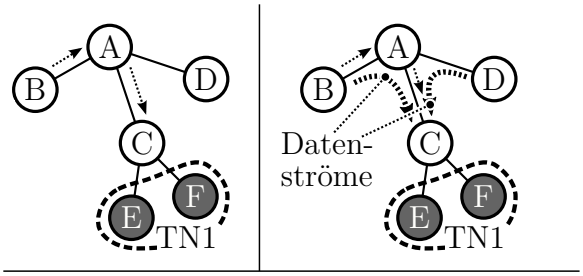


Abbildung 8.15: *Keepalive*-Übertragung ohne (l) und mit (r) Auslastung des Netzwerks

Eingangswarteschlangen ihrer Ports ab, und sortieren sie in die Ausgangswarteschlangen der entsprechenden Zielports ein. Dabei gelte die Annahme, dass die Switches schnell genug sind, um mit der vollen Datenrate zwischen allen Ports zu vermitteln, so dass die Vermittlung der Pakete zwischen den jeweiligen Warteschlangen nicht den Flaschenhals bildet. Es gelte zudem ein faires Weiterleitungsprinzip ohne Priorisierung. Ausgangspunkt der Betrachtung sei eine Anforderung von Knoten B, mit der er das Teilnetz der beiden ausgeschalteten Knoten E und F wecken möchte. Gibt es keinen anderen Netzwerkverkehr, wird die *Keepalive*-Nachricht von B durch Switch A direkt in die Ausgangswarteschlange des entsprechenden Ports gelegt und an C weitergeleitet, der daraufhin Knoten E und F weckt. Es entsteht keine nennenswerte Verzögerung. Gibt es auch anderen Netzwerkverkehr, beispielsweise eine Datenübertragung von Knoten D zu Knoten C, sortiert Switch A die Pakete von beiden Eingangswarteschlangen gleichberechtigt in die Ausgangswarteschlange des Links zu Knoten C ein. Solange die Datenrate der Übertragung verhältnismäßig niedrig ist und es keinen Rückstau von Paketen in der Eingangswarteschlange von Knoten C gibt, ist zu erwarten, dass jeweils nur wenige Pakete vor einer *Keepalive*-Botschaft im Puffer liegen. Bei einer maximalen Paketlänge von 1518 Byte und einer Übertragungsrate von 100 Mbit/s ergibt sich eine Übertragungszeit von ca. 122 Mikrosekunden, was eine weitaus geringere Größenordnung ist, als die Zykluszeit der *Keepalive*-Botschaften. Wie viele Pakete jeweils vor der *Keepalive*-Nachricht im Puffer liegen lässt, sich ohne Weiteres nicht sagen und hängt stark von den Eigenschaften des Datenstromes und der jeweiligen Realisierung der Weiterleitungsmechanik des Switches ab. Ethernet ohne Dienstgüte-Erweiterungen erlaubt keinerlei Zusicherungen oder Garantien, daher ist nur eine sachgerechte Abschätzung möglich. Unter der Voraussetzung, dass das Netzwerk nicht überlastet ist, sich also keine Staus in den Warteschlangen bilden, ist unter Berücksichtigung der obengenannten Erläuterungen jedoch nicht zu erwarten, dass sich maßgebliche Verzögerungen ergeben. Problematisch wird es, wenn Warteschlangen volllaufen, also wenn das Netzwerk überlastet ist. Dies kann passieren, wenn

mehrere Sender Datenströme mit hoher Datenrate an den selben Empfänger senden (z. B. B und D an C in Abbildung 8.15). Dann erlaubt Ethernet ohne Dienstgüte-Erweiterungen keine Aussage über auftretende Verzögerung bei der Zustellung einer *Keepalive*-Botschaft. Selbst der Verlust der Nachricht ist möglich, wenn beispielsweise die Ausgangswarteschlange von Knoten C voll ist. Die maximale Verzögerung der Botschaft (ohne Verlust) hängt von der Größe des Ausgangspuffers ab. Um in solchen Situationen die korrekte Funktion des Netzwerkmanagements zu gewährleisten, sind Dienstgüte-Erweiterungen notwendig. Eine Auswahl geeigneter Mechanismen soll im Folgenden kurz angesprochen werden.

8.10.2 Priorisierung

Ethernet-Switches unterstützen gemäß IEEE 802.1Q-2014 [41] Priorisierung bei der Weiterleitung von Ethernet-Frames. Im VLAN-Tag (Virtual Local Area Network) des Ethernet-Frames kann eine Priorität zwischen Null und Sieben für jeden Frame vergeben werden (PCP, Priority Control Point). Switches können statt einer Ausgangswarteschlange pro Port bis zu acht besitzen, welche auf die acht Prioritätsklassen abgebildet werden. Über einen Auswahlalgorithmus wird aus diesen Warteschlangen der nächste zu versendende Frame ausgewählt. Der einfachste Algorithmus (*Strict priority*, beschrieben in 802.1Q-2014 Abschnitt 8.6.8.1), wählt jeweils den nächsten Frame aus der höchstprioren, nicht leeren Warteschlangen zur Übertragung. Versieht man Netzwerkmanagement-Frames (und damit *Keepalive*-Pakete) mit der höchsten Priorität, werden sie anderen Datenströmen gegenüber bevorzugt und vor diesen weitergeleitet. Dies gewährleistet die Funktion des Netzwerkmanagements auch unter hohen Lastbedingungen.

8.10.3 Bandbreitenreservierung

Eine weitere Möglichkeit ist die Bandbreitenreservierung. Mit dem *Stream Reservation Protocol* (SRP) aus IEEE 802.1Q-2014 [41] kann auf allen Switches im Netzwerk Bandbreite für Datenströme reserviert werden. Dabei können (standardmäßig) 75 Prozent der verfügbaren Bandbreite für Datenströme entlang des gesamten Pfades von einer Quelle zu einer oder mehreren Senken reserviert werden [54]. Wird für alle Datenströme (nicht für das Netzwerkmanagement) Bandbreite reserviert, bleibt noch verfügbare Bandbreite außerhalb des reservierten Bereichs für das Netzwerkmanagement übrig.

Zusammen mit dem *credit based traffic shaping* aus IEEE 802.1Q-2014 ist dann sichergestellt, dass auch die Pakete aus nicht-reservierten Ausgangswarteschlangen zum Zuge kommen und versendet werden. Die Pakete aus reservierten Warteschlangen werden zeitlich verteilt, um eine konstante (die reservierte) Datenrate zu erzielen. In den Lücken zwischen diesen Paketen finden die übrigen 25 Prozent der Nachrichten ihren Durchsatz. Damit können zwar immer noch einige Pakete vor den Netzwerkmanagementbotschaften liegen und diese verzögern (die reservierten Warteschlangen werden bevorzugt), aber wie bereits beschrieben ist die Verzögerung durch einige, wenige Pakete für das Netzwerkmanagement nicht relevant. Matheus und Königseder beschreiben genau diesen Fall für sicherheitskritische Daten im Fahrzeug, die durch reservierte Ströme verzögert werden [54] können. Die andere Variante, Bandbreitenreservierung zu Nutzen, besteht darin, gezielt Bandbreite für den Netzwerkmanagementverkehr zu reservieren. Die durch das *Traffic Shaping* verursachten Verzögerungen der Pakete sind im Mikro- bis Millisekundenbereich [54] und für das Netzwerkmanagement irrelevant. Ein Problem dabei stellt jedoch die dynamische Reservierung dar, da Knoten (damit auch Switches) sich abschalten, und dann die Reservierungen vergessen. Eine Möglichkeit, dem zu begegnen, ist eine statische Reservierung, da die Kommunikationsanforderungen im Fahrzeug ohnehin bekannt sind. Dies wird auch in [54] vorgeschlagen, mit der Begründung, dass das Ausführen von SRP zum Systemstart zu lange dauert. Mit der Bandbreitenreservierung kann gewährleistet werden, dass der nötige Durchsatz der Netzwerkmanagement-Pakete stets gewährleistet ist. Natürlich ist dies nur sinnvoll, sofern das Netzwerk im Fahrzeug dies ohnehin benutzt.

8.10.4 Zeitgesteuertes Netzwerk

Die dritte Möglichkeit, Netzwerkmanagementpakete zugesichert weiterzuleiten, ist die Anwendung eines Zeitscheibenverfahrens im Netzwerk. Dann gibt es feste, regelmäßige Zeitintervalle für bestimmte Arten von Netzwerkbotschaften. Das zum Zeitpunkt der Erstellung der Arbeit in der Standardisierung befindliche Ethernet *Time Sensitive Networking* (TSN) beschreibt unter anderem ein solches Zeitscheibenverfahren für Ethernet. Die Arbeiten der *Time Sensitive Networking Task Force* [36] der IEEE 802.1-Arbeitsgruppe zielen darauf ab, Dienstgütemechanismen für zeit- und sicherheitskritische Datenübertragungen zu standardisieren. Unter dem Begriff *Time aware shaping* (IEEE 802.1Qbv) [42] wird ein zeitscheibenbasierter Kanal unter den ereignisgesteuerten Paketverkehr gemischt, so dass zu festgelegten, zyklischen Zeitfenstern nur kritische Datenpakete und keine sonstigen weitergeleitet werden [54]. Mittels dieser Fenster kann

ebenfalls zugesichert werden, dass die Pakete des Netzwerkmanagements rechtzeitig übertragen werden. Bei der Standardisierung von TSN sind auch zahlreiche Vertreter der Automobilindustrie involviert, so dass sehr wahrscheinlich ist, dass die daraus entstehenden Mechanismen für die Datenkommunikation im Automobil Verwendung finden werden. Unter dieser Voraussetzung kann das Problem der zugesicherten Dienstgüte für das Netzwerkmanagement als gelöst angesehen werden.

8.10.5 Fazit

Eine zugesicherte Übertragung von Netzwerkmanagement-Paketen wie *Keepalive* ist unabdingbar für eine korrekte Funktion des Netzwerks und von Teilnetzbetrieb. Einzelne Paketverluste oder geringfügige Verzögerungen von Paketen sind unkritisch. Solange keine Netzwerkverbindung überlastet ist, also sich keine Staus in den Warteschlangen der Switches bilden, sind die Verzögerungen durch andere Pakete vernachlässigbar. In Fahrzeugnetzwerken ist zu erwarten, dass jegliche Bandbreitenanforderungen von vornherein geplant sind [54], so dass es ohnehin nicht zu Überlastungen einzelner Verbindungen kommen sollte. Dennoch sind Zusicherungen möglich, wenn man Dienstgüteerweiterungen wie Priorisierung und Bandbreitenreservierungen aus IEEE 802.1Q hinzunimmt. Auch die Mechanismen aus Ethernet TSN werden sehr wahrscheinlich im Automobil zum Einsatz kommen. Zinner gibt in seiner Dissertation [86] einen Ausblick auf die Möglichkeiten, Dienstgütezusicherungen für Ethernet mit Blick auf den Einsatz im Fahrzeug zu gewährleisten. Insgesamt kann festgestellt werden, dass es bei der Fahrzeugbordvernetzung mittels Ethernet adäquate Möglichkeiten gibt, die rechtzeitige Übermittlung von Netzwerkmanagement-Paketen (und somit auch der *Multinet*-Botschaften) sicherzustellen.

8.11 Multinet und AUTOSAR

Multinet ist in das durch AUTOSAR standardisierte Netzwerkmanagement (vgl. Abschnitt 4.4) integrierbar. Die durch AUTOSAR definierten Netzwerkmanagement-Botschaften auf einem CAN-Bus können durch einen angebundenen *Multinet*-Knoten umgesetzt werden in *Keepalive*-Botschaften für das jeweilige Teilnetz, indem die durch die NM-Aggregation angeforderten *Partial Network Cluster* über die Schnittstelle von *Multinet* angefordert werden.

8.12 Zusammenfassung

Multinet stellt eine flexible und fehlertolerante Teilnetzbetrieb-Architektur dar, die eine Zustandskoordination beliebig partitionierbarer Teilnetze über ein verteiltes Netzwerkmanagement erlaubt. In den folgenden Kapiteln wird eine Implementierung von *Multinet* beschrieben, die als sowohl in Simulation zur Validierung des Konzeptes genutzt wurde, als auch in einem Testfahrzeug auf eingebetteter Hardware eingesetzt und unter realen Bedingungen validiert (Kapitel 11) wurde.

9 Implementierung von Multinet

9.1 Übersicht

Die hier beschriebene Implementierung von *Multinet* diente dazu, die technische Machbarkeit zu zeigen und daraus gewonnene Erkenntnisse in das Konzept zurückfließen zu lassen. Die Software ist modular und systemunabhängig aufgebaut. Systemspezifische Aufrufe sind über eine Laufzeitbibliothek für die jeweilige Plattform realisierbar, so dass *Multinet* in unterschiedlichen Hard- und Softwareumgebungen eingesetzt werden kann. Die Implementierung erfolgte in der Programmiersprache C. *Multinet* wurde in dem Simulationsframework OMNeT++ simuliert (vgl. Kapitel 10) und in einem Testfahrzeug unter realen Bedingungen eingesetzt (vgl. Kapitel 11). Das Komponentendiagramm in Abbildung 9.1 zeigt die Software-Struktur und die Abhängigkeiten der einzelnen Module.

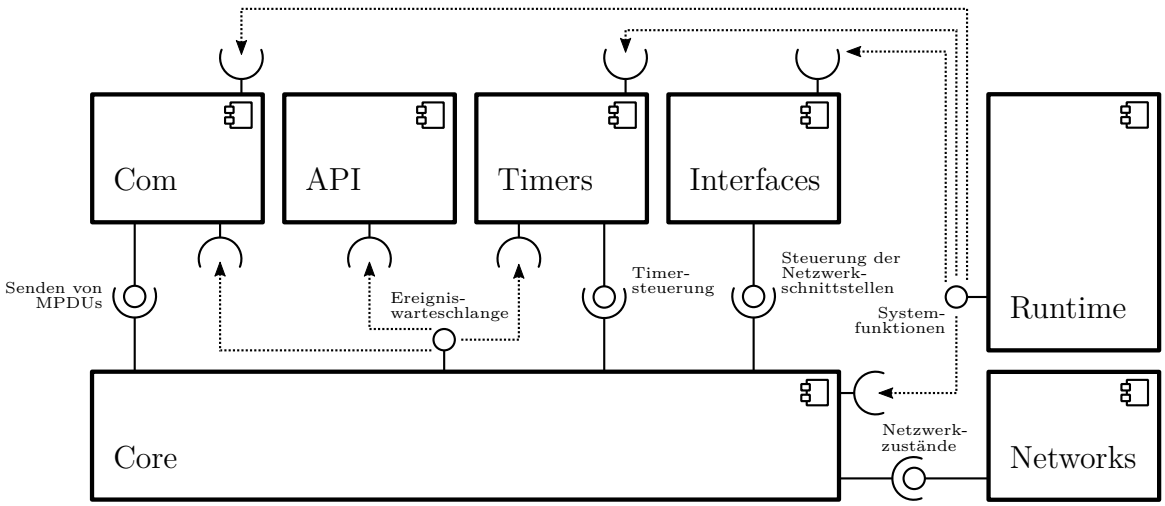


Abbildung 9.1: Software-Komponenten von *Multinet*

9.2 Kernkomponente (Core)

Die Kernkomponente implementiert den Zustandsautomaten von *Multinet* und verarbeitet Ereignisse, die von den jeweiligen Komponenten in eine Warteschlange gelegt werden. Die Warteschlange dient dabei zur Serialisierung und Pufferung von Ereignissen der verschiedenen Quellen. Die Realisierung der Warteschlange erfolgt innerhalb der Laufzeitumgebung (*Runtime*-Komponente), da hierfür betriebssystemeigene Mechanismen genutzt werden. Der Zugriff erfolgt durch Aufruf der entsprechenden Funktionen der Laufzeitkomponente. Für jedes Ereignis wird eine entsprechende Handler-Routine ausgeführt. Mögliche Ereignisse sind Timer-Abläufe, empfangene MPDUs und Aufrufe der Anwendungsschnittstelle.

9.2.1 Behandlung empfangener MPDUs

In der *Keepalive*-Botschaft sind zwei Informationen enthalten: Die Kennung des Netzwerks, welches wachgehalten werden soll, sowie die Priorität, die für das in Abschnitt 8.4.2 beschriebene Arbitrierungsverfahren verwendet wird.

Wird eine *Keepalive*-MPDU empfangen, wird die Netzwerk-TTL des durch die Kennung adressierten Teilnetzes in der Teilnetzverwaltung zurückgesetzt (vgl. Abschnitt 8.4.3). War das Teilnetz bislang inaktiv (also ist dies die erste *Keepalive*-Botschaft), wird es aktiviert (vgl. Abschnitt 9.2.4). Will der Knoten selbst das Teilnetz ebenfalls wachhalten, wird die *Keepalive*-Arbitrierung durchgeführt. Ist die Priorität der empfangenen Botschaft niedriger als die eigene, wird in der Teilnetzverwaltung der passive Wachhalte-Modus für das Netz aktiviert. Ist sie geringer, wird aktives Wachhalten konfiguriert. Beim Empfang einer *Query*-Botschaft wird wie in Abschnitt 8.5 für jedes Netzwerk, das der Knoten wachhält (aktiv oder passiv), eine asynchrone *Keepalive*-Botschaft gesendet.

9.2.2 Behandlung von Timern

Der Basistimer ist der primäre Taktgeber von *Multinet*. Mit jedem Ablauf wird ein Zyklus des Zustandsautomaten angestoßen und der Zustand der Netzwerkschnittstellen (*Interfaces*-Komponente) aktualisiert (vgl. Abschnitt 9.5). Zusätzlich wird die Netzwerk-TTL aller aktiven Teilnetze reduziert. Läuft die TTL eines Netzwerks ab, wird es deaktiviert (vgl. Abschnitt 9.2.4). Fällt die TTL eines Netzwerks, welches der

Knoten passiv wachhält, unterhalb des Erkennungsfensters, wird in der Teilnetzverwaltung aktives Wachhalten aktiviert und damit die Arbitrierungsphase gestartet (vgl. Abschnitt 8.4.2).

Bei jedem Ablauf des Keepalive-Timers (`T_KA`) wird für jedes Teilnetz, das der Knoten selbst aktiv wachhält (das mindestens eine lokale Anwendung angefordert hat) eine *Keepalive*-Botschaft versendet. Ereignisse des Timers `T_SLEEP` werden direkt als Eingangssignale des Zustandsautomaten gesetzt.

9.2.3 Behandlung von Aufrufen durch Anwendungen

Aufrufe der Anwendungsschnittstelle werden ebenfalls in die Ereigniswarteschlange eingereiht, da sie asynchrone Ereignisse aus einem anderen Thread-Kontext darstellen. Wird ein Teilnetz durch eine lokale Anwendung angefordert, und war es bislang inaktiv, wird es aktiviert (vgl. Abschnitt 9.2.4). Ist es bereits durch eine andere lokale Anwendung angefordert worden, und war bereits aktiv, wird lediglich ein Zähler in der Teilnetzverwaltung erhöht, der speichert wie viele Anwendungen dieses Netzwerk angefordert haben. Wird ein Teilnetz durch eine lokale Anwendung freigegeben, wird dieser Zähler einfach um eins verringert. Bei Ablauf des *Keepalive*-Timers werden nur für die Netzwerke *Keepalive*-Botschaften gesendet und die TTL zurückgesetzt, bei denen dieser Zähler größer null ist (also die durch mindestens eine lokale Anwendung angefordert werden) (vgl. Abschnitt 9.2.2). Der Knoten hört damit auf, das Netz wach zu halten, und sofern kein anderer Knoten es wach hält, läuft die TTL des Netzes automatisch auf allen Knoten ab und es wird deaktiviert (vgl. Abschnitt 9.2.4).

9.2.4 Aktivierung und Deaktivierung von Teilnetzen

Ein Teilnetz wird aufgrund von empfangenen *Keepalive*-MPDUs oder aufgrund einer Anforderung durch lokale Anwendungen aktiviert. Dabei wird der Zustand des Netzes in der Teilnetzverwaltung auf aktiv gesetzt. Ist der Knoten selbst Mitglied des Teilnetzes, wird der Zähler aktiver, eigener Netzwerke im Eingangsvektor des Zustandsautomaten um eins erhöht. Anschließend wird aus der Liste angeforderter Netzwerke in der Teilnetzverwaltung berechnet, welche direkten Nachbarn derzeit wach sein müssen, und dies an die *Interfaces*-Komponente gemeldet. Diese weckt entsprechend schlafende Nachbarn auf.

Ein Teilnetz wird deaktiviert, sobald dessen TTL auf null gefallen ist. Analog zum Aktivieren wird der Zustand des Netzes in der Teilnetzverwaltung auf inaktiv gesetzt, und der Zähler aktiver, eigener Netzwerke um eins reduziert. Ebenfalls wird der gewünschte Zustand der Nachbarn berechnet und an die *Interfaces*-Komponente gemeldet, die entsprechend das Herunterfahren der Verbindungen zu Nachbarn, die nicht mehr wach bleiben müssen, vorbereitet.

9.2.5 Zustandsautomat

Der Zustandsautomat ist wie in Abschnitt 8.7 beschrieben implementiert und durch den Basistimer `T_BASE` getaktet. Mit jedem Basistimer-Takt wird eine dem aktuellen Zustand entsprechende Routine ausgeführt, in der die Signale des Eingangsvektors ausgewertet werden, Aktionen ausgeführt werden und der nächste Zustand bestimmt wird.

9.3 Teilnetzverwaltung (Networks)

In der Teilnetzverwaltung werden Informationen über den aktuellen Zustand aller bekannten Teilnetze gespeichert. Tabelle 9.1 listet die verfügbaren Informationen auf. Über die Schnittstelle der Teilnetzverwaltung können die Einträge durch die Kernkomponente für jedes Netzwerk anhand der Netzkennung ausgelesen oder verändert werden.

9.4 Timerverwaltung (Timers)

Die Timerverwaltung bietet der Kernkomponente eine Schnittstelle zum Starten und Stoppen der benötigten Timer. Alle Timer werden von durch die Runtime-Komponente bereitgestellten System-Timer abgeleitet, der bei der Initialisierung konfiguriert wird. Eine übergebene Callback-Funktion wird bei Ablauf des Systemtimers aufgerufen und dekrementiert alle abgeleiteten Timer. Läuft einer der abgeleiteten Timer ab, wird ein entsprechendes Ereignis in die Warteschlange der Kernkomponente gelegt. Alle Timer sind über Konfigurationsparameter als Vielfache des Basistimers festlegbar (vgl. Anhang C). Der Basistimer selbst ist als Vielfaches des Systemtimers festlegbar.

Tabelle 9.1: Zustandsinformationen der Netzwerke in der Teilnetzverwaltung

Element	Name	Beschreibung
`network_id`	ID	Kennung des Netzwerks
`is_active`	Zustand	Sagt aus, ob das Netzwerk gerade aktiv oder inaktiv ist.
`port_mask`	Portzuordnung	Listet auf, über welche Ports der Knoten mit dem Teilnetz verbunden ist. Nachbarn an diesen Ports werden geweckt, wenn das Netz aktiviert wird.
`ttl`	Time-to-Live	Netzwerk-Timeout. Läuft die TTL ab, wird das Netzwerk deaktiviert.
`active_keepalive`	*Keepalive*-Modus	Sagt aus, ob das Netzwerk aktiv oder passiv wachgehalten wird.
`affiliated`	Netzzugehörigkeit	Gibt Auskunft, ob der Knoten selbst zu dem Teilnetz gehört.
`local_requests`	Anforderungen	Anzahl lokaler Anwendungen, die das Netzwerk gerade anfordern. Ist `local_requests` größer null, wird das Netzwerk vom Knoten wachgehalten.

9.5 Netzwerkschnittstellen (Interfaces)

Das *Interfaces*-Modul verwaltet den Zustand aller Netzwerkschnittstellen des Knotens und der Verbindungen zu seinen Nachbarn. Es bietet der Kernkomponente eine Schnittstelle um Nachbarn über das Netzwerk aufzuwecken und überwacht den Verbindungszustand zu allen Nachbarn. Die möglichen Zustände und Zustandswechsel jeder Netzwerkverbindung sind im Zustandsdiagramm in Abbildung 9.2 dargestellt. Ein Interface wird in zwei Fällen gestartet: Wird der entsprechende Nachbar durch die Kernkomponente angefordert, wird der Nachbar über das Netzwerk geweckt und die Schnittstelle aktiviert. Wird stattdessen ein Link-Signal auf dem Netzwerkmedium erkannt, bedeutet dies, dass ein vormals inaktiver Nachbar aufgewacht ist (Benachrichtigung über Callback-Methode durch Laufzeitkomponente und Schnittstellentreiber). In diesem Fall wird nur die Schnittstelle aktiviert, um die Verbindung herzustellen. Der Linkzustand wird kontinuierlich überwacht. Bricht die Verbindung ab, wird ein erneuter Verbindungsversuch (Rücksetzen der Schnittstelle) probiert. Wird ein Interface (und damit ein Nachbar) von der Kernkomponente nicht mehr angefordert, darf die Netzwerkverbindung nicht einfach beendet werden, denn damit würde der noch wache Nachbarknoten vom Netzwerk abgeschnitten. Außerdem wäre seine eigene Netzwerkschnittstelle noch aktiv, weswegen sofort ein Link-Signal erkannt werden und das

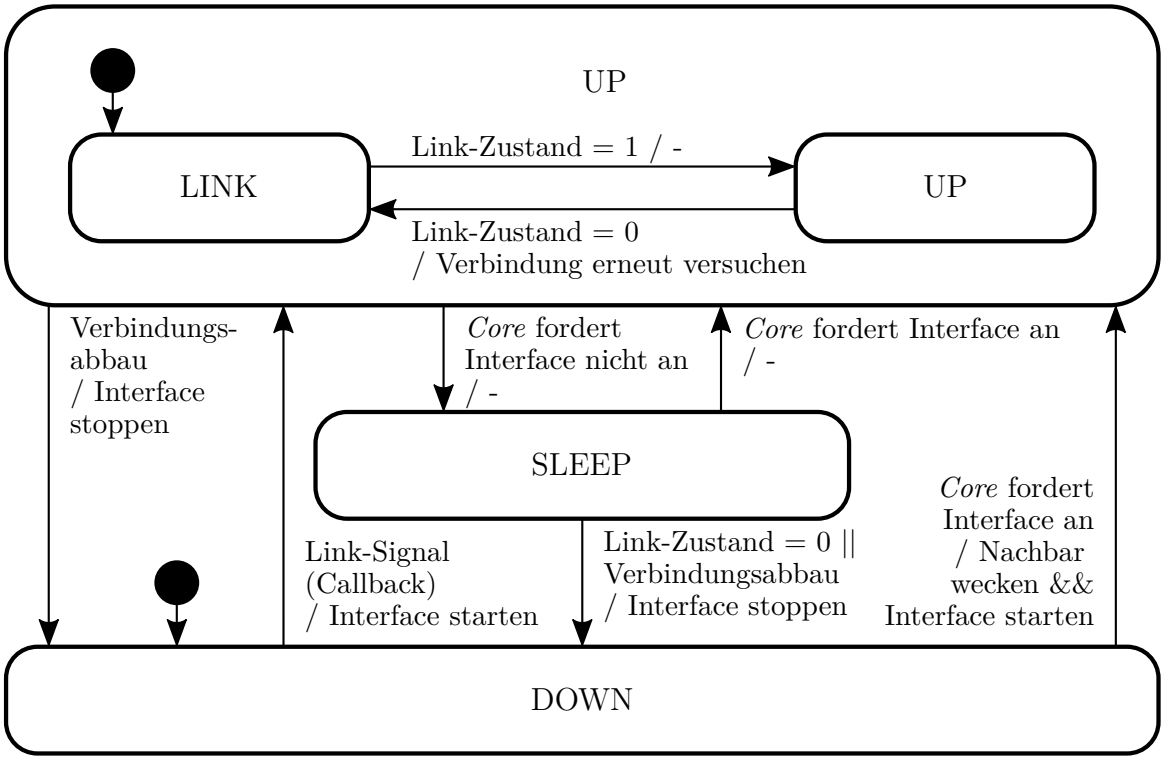


Abbildung 9.2: Zustandsdiagramm der Netzwerkschnittstellen und -verbindungen

Interface wieder neu gestartet werden würde. Zur Synchronisation beim Herunterfahren des Nachbarknotens gibt es den Zwischenzustand *SLEEP*. In diesem Zustand wird darauf gewartet, dass der Nachbarknoten sich abschaltet und sein eigenes Interface deaktiviert. Nach Verlust der Verbindung wird anschließend auch das eigene Interface abgeschaltet. Das *Interfaces*-Modul greift zur Konfiguration der Netzwerkschnittstellen auf entsprechende Funktionen der Laufzeitkomponente zu.

9.6 Kommunikationsmodul (Com)

Das *Com*-Modul versendet und empfängt MPDUs. Es bietet der Kernkomponente eine Schnittstelle um *Keepalive*- und *Query*-Botschaften zu senden und legt empfangene Nachrichten als Ereignisse in die *Core*-Warteschlange. Die Laufzeitkomponente bietet hierzu dem *Com*-Modul Funktionen zum Senden und Empfangen von Netzwerkpaketen (die z. B. bei Vorhandensein einer Berkeley Sockets-Implementierung entsprechend deren Funktionen kapseln). Je nach zugrundeliegendem System kann der Empfang von Botschaften blockierend in einem eigenen Thread erfolgen, oder durch Aufruf (Callback) durch die Laufzeitkomponente.

Tabelle 9.2: Funktionen der Anwendungsschnittstelle

Funktion	Parameter	Beschreibung
`request_network`	Netzwerkkennung	Netzwerk wird durch Anwendung angefordert.
`suspend_network`	Netzwerkkennung	Angefordertes Netzwerk wird durch Anwendung wieder freigegeben.

9.7 Laufzeitumgebung (Runtime)

Die Laufzeitkomponente (*Runtime*, RT) bettet die plattformunabhängigen Module von *Multinet* in das darunterliegende System ein. Sie abstrahiert systemspezifische Funktionen, wie das Starten von System-Timern, die Realisierung einer Nachrichtenwarteschlange, das Konfigurieren von Netzwerkschnittstellen und das Senden und Empfangen von Netzwerkpaketen. Das Laufzeitmodul ist die einzige Komponente, die für jedes System, welches *Multinet* einbindet, portiert werden muss. Im Rahmen der vorliegenden Arbeit wurde die RT für das Simulationsframework OMNeT++ [79] (vgl. Kapitel 10) und für das Betriebssystem eCOS (*embedded Configurable Operating System*) (vgl. Kapitel 11) implementiert.

9.8 Anwendungsschnittstelle (API)

Die Anwendungsschnittstelle ermöglicht es anderen Softwarekomponenten, Teilnetze anzufordern. Die asynchronen Anforderungen und Freigaben werden als Ereignisse in die Warteschlange der Kernkomponente gelegt und von ihr verarbeitet. In der Teilnetzverwaltung (vgl. Abschnitt 9.3) werden für jedes Netzwerk die Anforderungen und Freigaben mehrerer Applikationen durch den Zähler `local_requests` aggregiert. Ist dieser Wert größer Null, wird das Netzwerk durch *Multinet* wachgehalten.

10 Simulation von Multinet

10.1 Übersicht

Die im vorangegangenen Kapitel 9 beschriebene Implementierung von *Multinet* wurde im Rahmen einer umfassenden Simulation validiert und gegen das ursprüngliche Konzept getestet. Die damit gewonnenen Daten lassen eine Analyse und Veranschaulichung des Zeitverhaltens von Zustand, Paketverkehr und Netzwerkauslastung zu. Die Simulation ermöglicht somit eine gezielte Untersuchung verschiedener Beispielnetze und -situationen, um Kernfunktionen, das Verhalten im Fehlerfall oder Skalierungs- und Gleichzeitigkeitseffekte zu beobachten und konzeptuelle und implementierungsspezifische Fehler zu erkennen und zu beheben.

Als Grundlage wurde das Simulationsframework OMNeT++ (*Objective Modular Network Testbed in C++*) [79] verwendet, welches im folgenden Abschnitt kurz beschrieben wird. OMNeT++ wird in [80] ausführlich beschrieben. Die Codebasis von *Multinet* wurde an OMNeT++ angebunden und eine Reihe unterschiedlicher Szenarien simuliert. Die aufgezeichneten Daten wurden ausgewertet und visuell aufbereitet.

10.2 Das Simulationsframework OMNeT++

OMNeT++ ist ein diskretes, ereignisbasiertes Simulationsframework, mit dem vornehmlich Netzwerke und Protokolle simuliert und getestet werden können. Das Framework wurde von A. Varga in [80] vorgestellt und detailliert beschrieben. An dieser Stelle soll ein kurzer Überblick gegeben werden, um nachvollziehbar zu machen, auf welche Weise und mit welchen Mitteln die folgenden Daten und Auswertungen erzeugt wurden.

10.2.1 Modellierung von Netzwerken und Protokollen

Ein Modell in OMNeT++ setzt sich aus diskret miteinander verbundenen Modulen zusammen, die einander und sich selbst Nachrichten (Ereignisse) zu einer planbaren Zeit zustellen. Ein Modul wird durch eine C++-Klasse beschrieben und muss zumindest die Funktionen `initialize()` und `handleMessage()` besitzen, die durch den Simulationskernel aufgerufen werden (bei Simulationsstart, respektive bei Auftreten eines zuvor geplanten Ereignisses bzw. einer Nachricht). Ereignisse lassen sich in der simulierten Zeit genau planen und zustellen, wodurch beispielsweise Timer nachgestellt oder Netzwerkbotschaften über Verbindungen an benachbarte Knoten gesendet werden können. Ist die geplante Simulationszeit eines Ereignisses oder einer Nachricht gekommen, wird die `handleMessage()`-Methode des Empfängermoduls aufgerufen, in der die Nachricht (bzw. das Ereignis) behandelt werden kann. Zur Verbindung mit Nachbarn verfügen Module über sogenannte *Gates*, welche die Endpunkte der Verbindungen darstellen und damit in etwa Netzwerkports oder Verbindungssteckern bei realen Systemen nachempfunden sind. Module lassen sich zu hierarchischen Verbundmodulen (*Compound Modules*) zusammensetzen, wodurch komplexe Systeme modelliert werden können. Auf diese Weise wurden durch die Community bereits zahlreiche Netzwerkprotokolle und Protokollfamilien, sowie Netzwerkapplikationen und verteilte Systeme simuliert. Ein gutes Beispiel hierfür ist das *INET*-Framework, in dem eine große Anzahl an Protokollen und Schichten der TCP/IP-Protokollfamilie nachgebaut wurde. Netzwerke schließlich bilden die oberste Einheit in der OMNeT++-Modellhierarchie. Sie bestehen aus vernetzten Modulen und Verbundmodulen.

Die Module, Verbundmodule und Netzwerke werden in einer eigenen Beschreibungssprache, NED, definiert. Für jedes Modul gibt es eine NED-Datei, welche Gates und Parameter des Moduls beschreibt, sowie eine C++-Klasse, die das Verhalten des Moduls implementiert. Bei Verbundmodulen beschreibt die NED-Datei die Zusammensetzung aus mehreren Kind-Modulen. Netzwerke sind in NED durch die vernetzten Module und ihre Verbindungen untereinander beschrieben.

Parameter können in NED definiert und spezifisch für verschiedene Simulationen per .ini-Datei festgelegt werden. Aus der C++-Implementierung des Moduls kann lesend auf die Parameter zugegriffen werden. Dies erlaubt eine Umkonfiguration von Modulen für verschiedene Simulationsläufe ohne den Quellcode zu verändern und neu kompilieren zu müssen.

10.2.2 Simulationen

Die OMNeT++-Modelle können anschließend simuliert werden. Die Simulation läuft dabei nicht in Echtzeit, sondern ereignisbasiert. Auftretende Ereignisse werden an das oder die Zielmodule durch Aufruf der `handleMessage()`-Methode der jeweiligen Instanz zugestellt, der entsprechende Code ausgeführt und daraus die nächsten Ereignisse geplant. Anschließend springt die simulierte Zeitachse zum nächsten, anstehenden Ereignis. Die Simulation endet nach einer festgelegten (Simulations-)Zeit oder wenn keine weiteren Ereignisse mehr anstehen. In einer zur Simulation gehörigen .ini-Datei werden zentral Modulparameter und Einstellungen wie Simulationsdauer, etc. festgelegt.

OMNeT++ bietet einen einfachen Weg, Daten, Kennzahlen und Metriken zu erzeugen und während der Simulation zu exportieren. Sogenannte Signale können durch die Module an beliebigen Stellen im Code emittiert und als Vektoren über die Zeitachse exportiert oder als Skalare berechnet werden. Die Ergebnisvektoren können anschließend mit einer eigenen grafischen Oberfläche ausgewertet und visualisiert werden oder mit dem *scavetool* des Frameworks gefiltert und in .csv-Dateien zur weiteren Aufbereitung exportiert werden.

10.2.3 Grafische Oberfläche

Die grafische Oberfläche von OMNeT++ (*tkenv*) dient der Veranschaulichung von Netzwerken, Modulen und Nachrichten. Module sind als (konfigurierbare) Icons dargestellt, Verbindungen als Linien dazwischen. Ereignisse wie die Übertragung von Nachrichten über eine Verbindung sind animiert. Knoten, Verbindungen und Nachrichten können per Mausklick genauer inspiziert werden. Die Visualisierung ist aus dem Quellcode heraus modifizierbar, so dass beispielsweise Texte geschrieben oder Symbole verändert werden können, was beispielsweise zur Visualisierung der Zustände der *Multinet*-Knoten genutzt wurde.

10.3 Einbindung von Multinet

Ein Ziel der Einbindung von *Multinet* in OMNeT++ war, dass die Codebasis unverändert bleiben sollte, um den tatsächlichen Quellcode und Änderungen daran di-

rekt testen und validieren zu können. *Multinet* ist in C geschrieben und damit nicht objektorientierter Natur, im Gegensatz zu OMNeT++. Der statische Zustand von *Multinet* (z. B. Timerwerte, Schnittstellen- und Netzwerkzustände, Zustandsmaschine) gilt daher prozessweit, was im Widerspruch dazu steht, dass in OMNeT++ alle Netzwerkknoten als Instanzen einer Klasse im selben Prozesskontext existieren. Um dieses Problem zu umgehen, wird der gesamte Zustand als Instanzvariable jedes Knotens gespeichert und bei der Ausführung der `handleMessage()`-Methode jeweils global wieder hergestellt. Der plattformunabhängige Quellcode von *Multinet* wurde nicht verändert. Für OMNeT++ wurde eine passende Laufzeitkomponente als Basis für den plattformunabhängigen Teil geschrieben. Damit wurde beispielsweise das Senden und Empfangen von MPDUs durch OMNeT++-Nachrichten realisiert und der Basistimer durch ein zyklisches Ereignis, welches sich das Modul selbst zustellt. Die Funktionen der Laufzeitkomponente wurden als Klassenmethoden realisiert, die über Funktionspointer mit zusätzlichem Instanzpointer-Argument aus *Multinet* aufgerufen werden. *Multinet* konnte somit vollständig und unverändert in die OMNeT++-Umgebung integriert werden.

10.3.1 Das OMNeT++-Modul MultinetNode

Zur Simulation von Netzwerken wurde ein OMNeT++-Modul implementiert, welches das Verhalten eines eingebetteten Netzwerkgerätes simuliert und den *Multinet*-Dienst einbindet.

Ein einfaches Zustandsmodell (vgl. Abbildung 10.1) beschreibt das grundlegende Verhalten des Moduls. Ein Knoten kann durch die Klassenmethode `MultinetNode::wakeUp()` durch einen Nachbarn (oder durch sich selbst) aufgeweckt werden. Dabei wird ein *Boot-Ereignis* (in Form einer OMNeT++-Nachricht an den Knoten selbst) nach Ablauf einer festgelegten Boot-Zeit geplant. Diese ist als Modulparameter festlegbar und beträgt standardmäßig 100 ms. Sobald das Ereignis eintritt (also die Boot-Zeit abgelaufen ist), gilt der Knoten als aktiv, das Symbol in der Visualisierung wird verändert und *Multinet* gestartet. Schaltet sich der Knoten ab (*Multinet* initiiert das Herunterfahren über die Funktion `rt_shutdown()` aus der Laufzeitkomponente), läuft dies nach demselben Schema ab. Auch die Dauer des Herunterfahrens ist parametrierbar und liegt standardmäßig bei 10 ms. Durch diese beiden Zeitparameter wird das Zeitverhalten eines echten Gerätes beim Ein- und Ausschalten simuliert. Neben den Methoden zur Modellierung des Zustandes und der Laufzeitkomponente

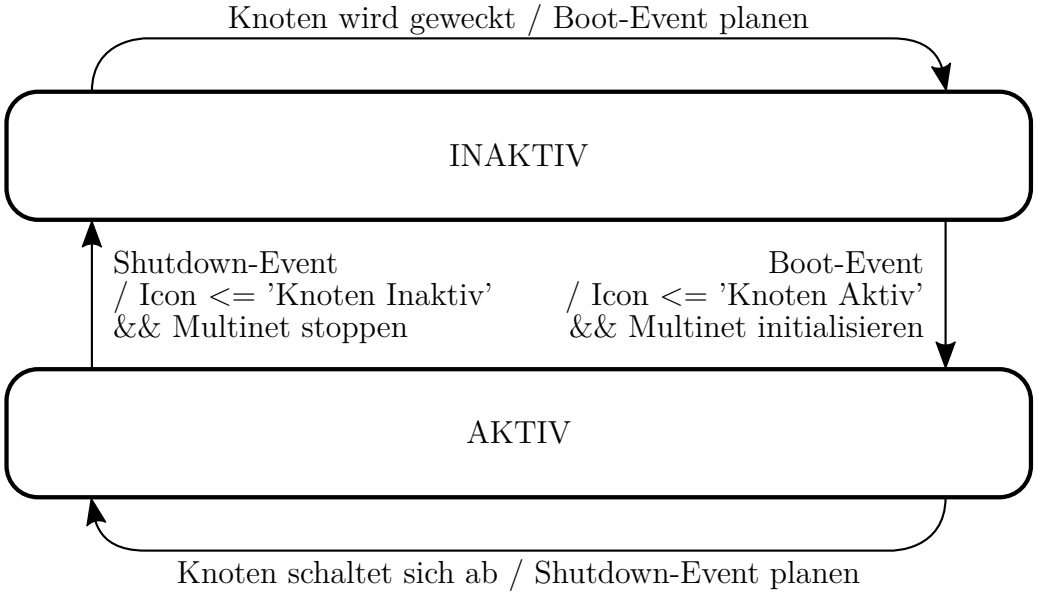


Abbildung 10.1: Modellierung des Knotenzustandes des MultinetNode-Knotens

enthält das Modul Methoden zur Konfiguration der Netzwerkliste, und eine Anwendung, die *Multinet* nutzt, um aus einer Konfigurationsdatei auszulesen, zu welchem Zeitpunkt sie welche Netzwerke anfordern und freigeben soll.

10.3.2 Konfiguration der Teilnetztabelle

Über einen Modulparameter wird für jeden Knoten festgelegt, in welchen Netzwerken er sich befindet. Dies erfolgt in Form eines Bitvektors, bei dem die Bitposition der jeweiligen Netzwerkkennung entspricht. Bei der Initialisierung liest jeder Knoten rekursiv den Netzwerk-Vektor aller Nachbarknoten und aller von ihnen ausgehenden Ästen des Netzwerks aus. Er erlernt somit alle vorhandenen Teilnetze und die zugehörigen Ports (*gates*), um damit die Teilnetztabelle zu erstellen. Dies entspricht einer einfachen Variante des automatischen Erlernens der Netztopologie. Nach der Initialisierung der Simulation besitzt jeder MultinetNode eine vollständige Teilnetztabelle. Damit muss lediglich der Netzwerk-Vektor jedes Knotens von Hand festgelegt werden, der Rest wird automatisch erkannt.

10.3.3 Simulierte Anforderung und Freigabe von Netzwerken

Um beliebige Situationen darstellen und simulieren zu können, kann jedem MultinetNode eine Konfigurationsdatei zugewiesen werden, in der Zeitpunkte (in Millisekunden)

und für jeden Zeitpunkt ein Bitvektor enthalten sind. Der Bitvektor gibt Auskunft, welche Netzwerke zu dem entsprechenden Zeitpunkt durch den Knoten angefordert werden sollen, und welche nicht. Bei der Initialisierung wird diese Datei eingelesen und es werden entsprechende Ereignisse zu den jeweiligen Zeitpunkten geplant. Damit kann für jeden Knoten individuell eine Art Handlungsstrang festgelegt und damit jedes beliebige Szenario aus nebenläufigen Anforderungen von Netzen nachgestellt werden.

10.3.4 Visualisierung

Die Visualisierung der *Multinet*-Netzwerke in OMNeT++ erfolgt über das graphische Frontend *tkenv*. Zusätzlich zur standardmäßigen Visualisierung wird die Farbe des Icons jedes Knotens entsprechend seinem Zustand verändert (hell: an; dunkel: aus). Über jeden Knoten wird zudem der aktuelle Zustand von *Multinet* (vgl. Abschnitt 8.7) geschrieben. Außerdem werden zwei Listen mit Netzwerkkennungen über jedem Knoten dargestellt. $N = \{...\}$ enthält die Netzkennungen der Netze in denen der jeweilige Knoten selbst Mitglied ist, $R = \{...\}$ diejenigen, die der Knoten gerade anfordert. Abbildung 10.2 zeigt die Visualisierung eines Beispielnetzwerks. Knoten A hat gerade Teilnetz 8 angefordert und bereits Knoten B (Zustand REMOTE WAKE) geweckt. Die anderen Knoten sind noch inaktiv.

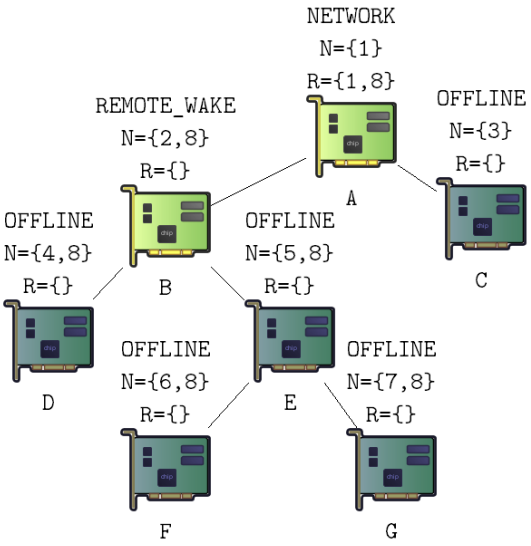


Abbildung 10.2: Grafische Darstellung von *Multinet* in OMNeT++/*tkenv*

10.4 Aufzeichnung und Darstellung von Daten

Zur Auswertung werden mit Hilfe der weiter oben beschriebenen Möglichkeit von OMNeT++, Daten mitzuschreiben und zu exportieren, folgende Vektoren kontinuierlich aufgezeichnet.

- Zustand des *Multinet*-Kerns. Dieser ermöglicht die Aufzeichnung der Aktivität des Knotens über die Zeit.
- Zeitpunkte gesendeter und empfangener Pakete. Ermöglicht die Aufzeichnung des zeitlichen Auftretens von *Keepalive* und *Query*-Paketen und die Betrachtung der verursachten Netzwerk- bzw. Interrupt-Last.
- Netzwerkanforderungen und -freigaben. Ermöglicht die Darstellung der Anforderungs- und Freigabezeitpunkte der verschiedenen Teilnetze und die zeitliche Korrelation der anderen Daten dazu.

Mit diesem Datensatz lässt sich das Verhalten von *Multinet* in beliebigen Netzwerktopologien und Situationen beobachten und auswerten. Die Daten aus dem Ergebnisvektor werden nach der Simulation automatisiert in eine Reihe von .csv-Dateien exportiert, die anschließend durch das Plot-Programm *gnuplot* grafisch dargestellt werden.

10.5 Simulationen

In diesem Abschnitt werden verschiedene, in OMNeT++ simulierte Beispielsituationen vorgestellt, die Ergebnisse veranschaulicht und das Verhalten von *Multinet* dabei diskutiert. Für alle Simulationen findet sich in Anhang D das zugehörige Simulationsprotokoll mit dem Zeitverlauf aller Knotenzustände. Bei den ersten, grundlegenden Simulationen wird dieses zur Veranschaulichung auch im jeweiligen Kapitel selbst dargestellt. In den folgenden Zeitdiagrammen werden die *Multinet*-Zustände OFFLINE (1), BRIDGE (2), REMOTE WAKE (3), NO NETWORK (4), NETWORK (5) der jeweiligen Knoten im zeitlichen Verlauf dargestellt.

10.5.1 Anfordern und Freigeben eines Teilnetzes

In der ersten Simulation wird das einfache Anfordern eines Teilnetzes, das Aufwecken der Knoten, das Freigeben des Netzes und schließlich das Einschlafen der Knoten validiert. Als Beispieltopologie dient das in Abbildung 10.3 dargestellte Netzwerk. Knoten A bis D stellen jeweils ein eigenes Teilnetz 1 bis 4 dar, zusätzlich bilden Knoten C und D zusammen ein gemeinsames Teilnetz 5. Knoten A soll Teilnetz 5 (Knoten C und D) anfordern, eine gewisse Zeit lang wachhalten, und anschließend wieder freigeben. Im ersten Beispiel ist Knoten B zu Beginn bereits wach, es müssen also nur Knoten C und D geweckt werden. Zu bemerken ist, dass B selbst nicht Teil des angeforderten Teilnetzes ist.

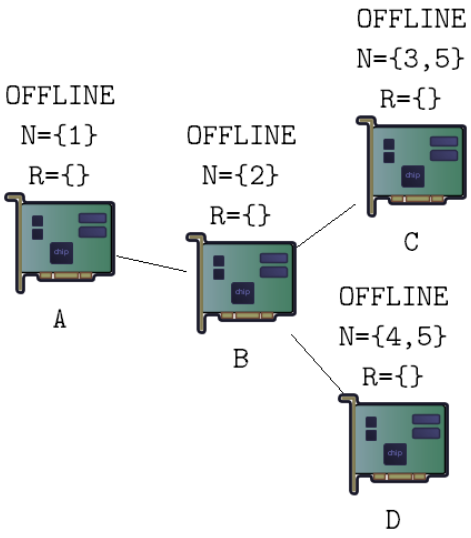


Abbildung 10.3: Einfaches *Multinet*-Netzwerk mit vier Knoten

Der Verlauf des Knotenzustandes aller vier Knoten ist für die erste Simulation in Abbildung 10.4 dargestellt, Abbildungen 10.5 und 10.6 zeigen das Netzwerk zu ausgewählten Zeitpunkten. Zunächst fordert Knoten A Teilnetz 5 an (1150 ms, 10.5a), und verschickt eine *Keepalive*-MPDU. B empfängt sie und weckt daraufhin seine beiden Nachbarn C und D, die im angeforderten Teilnetz liegen. 150 ms später sind beide gestartet und befinden sich im Zustand REMOTE WAKE (1300 ms, 10.5b). Beide versenden ein *Query*-Paket, im Zeitdiagramm 10.4 ist die daraufhin von A verschickte, asynchrone *Keepalive*-Botschaft zu sehen. Ab diesem Zeitpunkt wissen C und D, dass Teilnetz 5 angefordert wird und wechseln zum Zustand NETWORK REQUESTED. Knoten A sendet nun für die folgenden zwei Sekunden *Keepalive*-Botschaften und gibt das Teilnetz danach wieder frei, wie im Zeitdiagramm markiert. In den folgenden zwei Sekunden (vier *Keepalive*-Zyklen) läuft die TTL von Teilnetz 5 ab und Knoten C und D wechseln in den Zustand NO NETWORK REQUESTED (5110 ms, 10.6a). Eine Zykluszeit später (5600 ms) wechseln beide Knoten in den Zustand OFFLINE und schalten sich nach kurzer Nachlaufzeit aus (6160 ms, 10.6b). Durch diesen Testfall konnte das Anfordern und Freigeben von Teilnetzen über die Anwendungsschnittstelle, die Funktionsfähigkeit des Zustandsautomaten, asynchrone und zyklische *Keepalive*-Botschaften, die Zustandskoordination von Netzen, das Wecken von Nachbarn sowie das Abschalten von Teilnetzen und Knoten validiert werden.

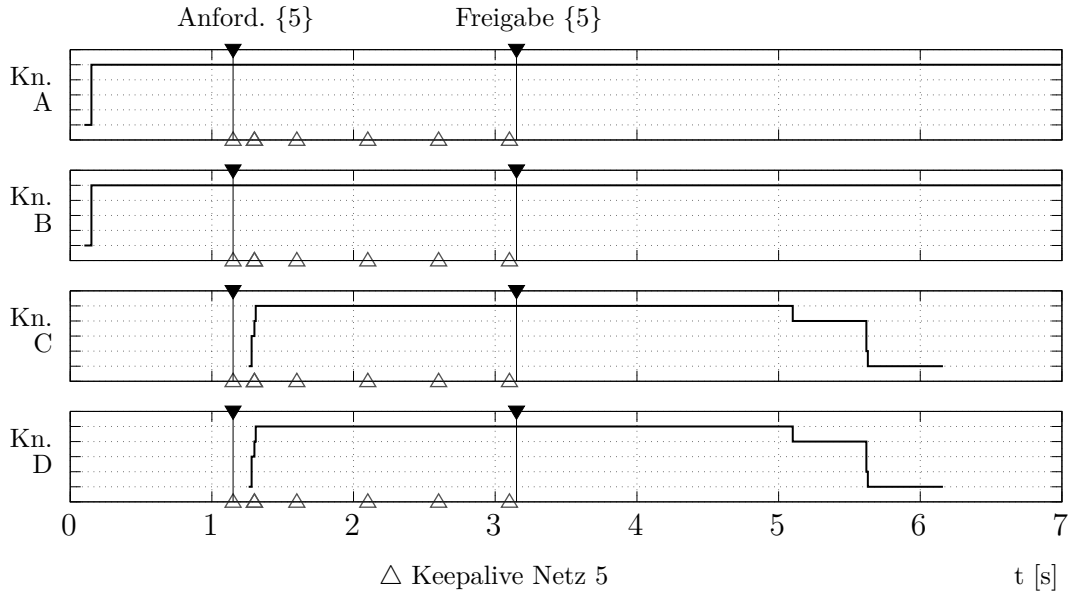
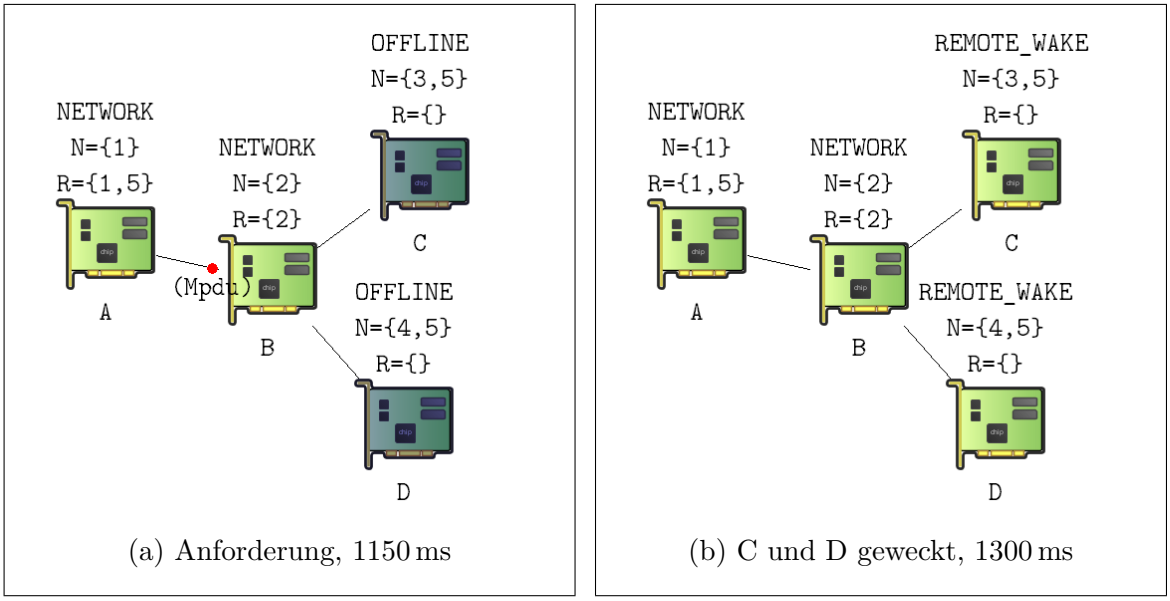


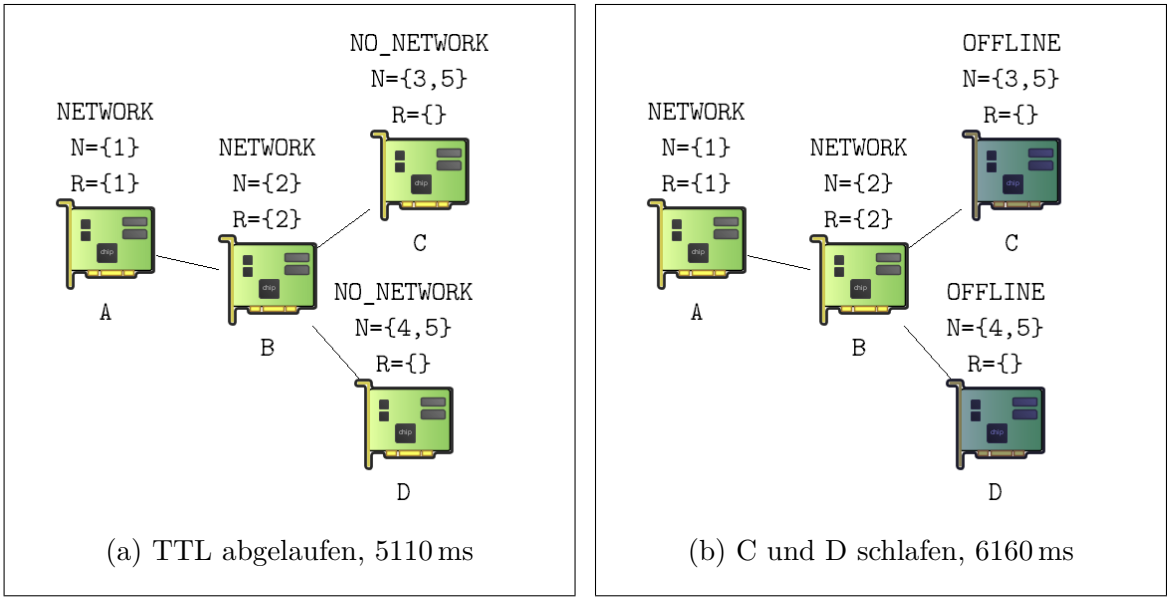
Abbildung 10.4: Zustandsverlauf beim Anfordern von Teilnetz 5



(a) Anforderung, 1150 ms

(b) C und D geweckt, 1300 ms

Abbildung 10.5: Anfordern von Teilnetz 5 durch Knoten A



(a) TTL abgelaufen, 5110 ms

(b) C und D schlafen, 6160 ms

Abbildung 10.6: Einschlafen des Teilnetzes 5

10.5.2 Anforderung mit schlafendem Knoten B

Bei diesem Simulationslauf ist Knoten B zu Beginn der Simulation abgeschaltet (vgl. Abbildung 10.8a). Er ist auch nicht im angeforderten Teilnetz 5, liegt aber zwischen dem anfordernden Knoten A und dem Teilnetz. Abbildung 10.7 zeigt den Verlauf aller Knotenzustände im Zusammenhang. Nach der Anforderung des Teilnetzes (1150 ms, 10.8a) weckt Knoten A seinen Nachbarn B (1280 ms, 10.8b), der zunächst ein *Query*-Paket sendet. A antwortet mit einem asynchronen *Keepalive* für Teilnetz 5, woraufhin B Knoten C und D weckt (1430 ms, 10.9a). Diesmal sind beide knapp 300 ms nach Anforderung im Zustand REMOTE WAKE (vgl. 150 ms vorher). Hier ist das *Hop-by-Hop*-weise Aufwachen eines Teilnetzes zu sehen. Knoten B befindet sich jedoch in keinem angeforderten Teilnetz, bleibt daher für die Zeitdauer T_SLEEP in NO NETWORK REQUESTED und wechselt dann zu BRIDGE. Da B nun zwei Netzsegmente brückt, darf er sich nicht abschalten, sondern muss im Zustand BRIDGE bleiben (vgl. Abbildung 10.9b). Dadurch bleibt der Pfad zwischen Knoten A und dem angeforderten Teilnetz 5 erhalten. Nachdem A Teilnetz 5 wieder freigegeben hat und dessen TTL abgelaufen ist, schlafen Knoten C und D zunächst wie im vorherigen Beispiel ein. Danach verlässt B den Zustand BRIDGE und darf sich selbst abschalten. Dieses Beispiel erweitert die getesteten Funktionen aus der vorherigen Simulation um die Validierung des *Hop-by-Hop*-Weckens mehrerer Knoten und um die konzeptgemäße Funktion des BRIDGE-Zustandes.

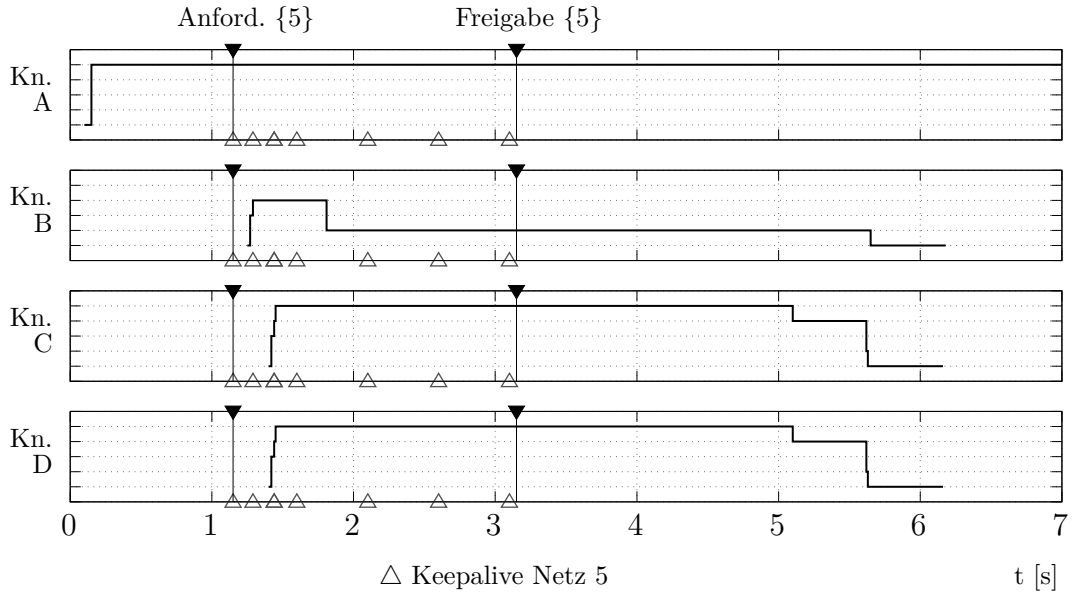
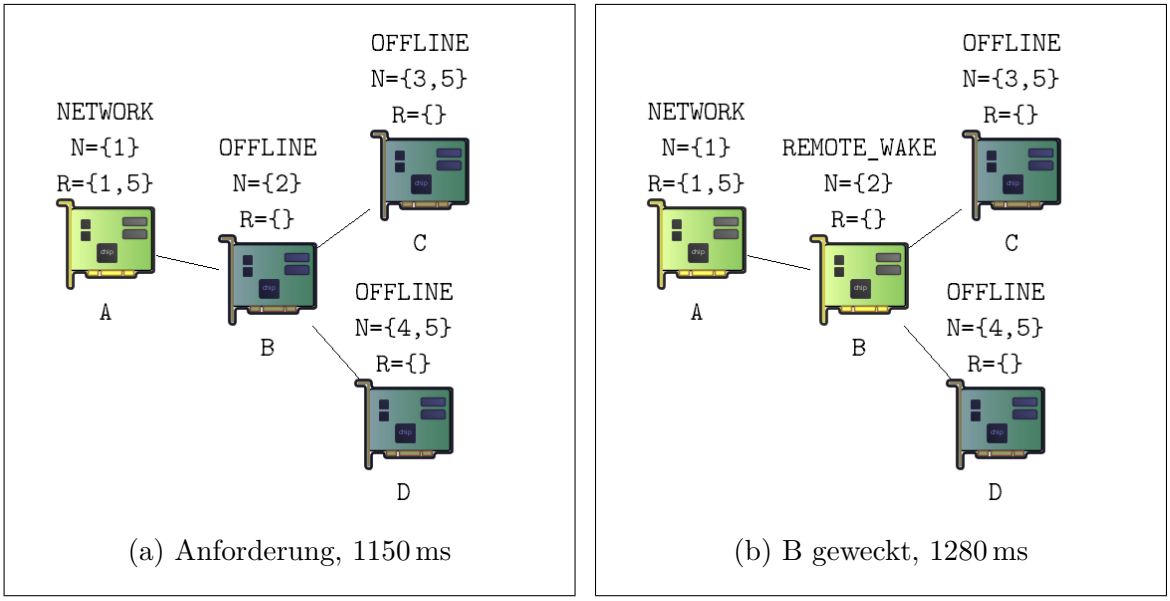


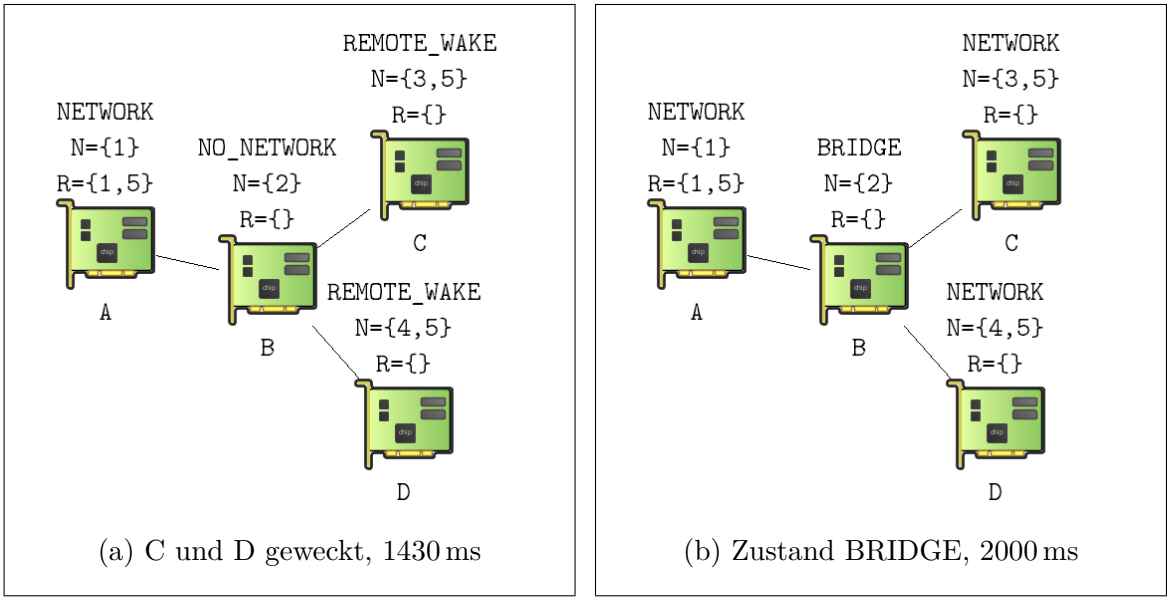
Abbildung 10.7: Anfordern von Teilnetz 5 bei schlafendem Knoten B



(a) Anforderung, 1150 ms

(b) B geweckt, 1280 ms

Abbildung 10.8: Anfordern von Teilnetz 5 und Wecken von Knoten B



(a) C und D geweckt, 1430 ms

(b) Zustand BRIDGE, 2000 ms

Abbildung 10.9: Brücken des Netzwerks durch den Knoten B

10.5.3 Wachhalten durch mehrere Knoten gleichzeitig

In diesem Beispiel wird das gleichzeitige Wachhalten von Teilnetzen durch mehrere Knoten simuliert. Zunächst fordert Knoten A Teilnetz 4 an (Knoten D). Später fordert Knoten C dasselbe Teilnetz an. A wurde für diese Simulation mit einer geringeren *Keepalive*-Priorität parametriert als C. Die Ergebnisse sind in gewohnter Form in Abbildung 10.10 dargestellt. Mit der Anforderung des Teilnetzes wird Knoten D aufgeweckt und durch die *Keepalive*-Nachrichten von Knoten A wachgehalten. Sobald auch Knoten C das Teilnetz anfordert, sendet er zunächst eine asynchrone *Keepalive*-Botschaft ins Netz. Diese empfängt auch Knoten A und wechselt in den passiven Wachhalte-Modus (d.h. stellt das Senden eigener *Keepalive*-Pakete ein, vgl. Kapitel 8.4.2), da seine Priorität geringer ist als die in der Nachricht von A codierte. Für das folgende Zeitintervall hält nun Knoten C das Teilnetz aktiv wach. Nachdem C es wieder freigibt, verstreichen zunächst zwei *Keepalive*-Perioden ohne gesendete Botschaft (das Erkennungsfenster), woraufhin Knoten A wieder in den aktiven Wachhaltemodus wechselt und das Senden von *Keepalive*-Nachrichten fortsetzt. Damit validiert dieser Testfall das *Keepalive*-Arbitrierungsverfahren, das Wechseln zwischen aktivem und passivem Wachhaltemodus und den konsistenten Anforderungszustand des Teilnetzes während der Übernahme des aktiven Wachhaltens durch einen anderen Knoten.

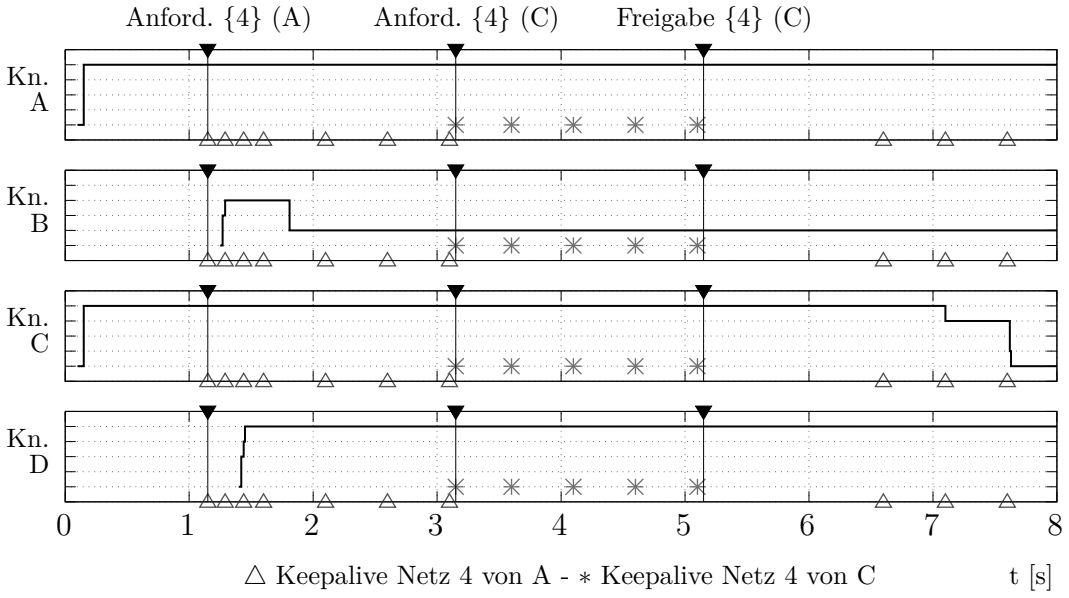


Abbildung 10.10: Gleichzeitiges Wachhalten von Teilnetz 4 durch mehrere Knoten

10.5.4 Wecken eines größeren Teilnetzes

In dieser Simulation wird das Wecken eines größeren Teilnetzes mit mehreren Kaskadierungsebenen veranschaulicht. Dabei wird das in Kapitel 8, Abschnitt 8.5 beschriebene Beispiel (vgl. Abbildung 8.5) nachgestellt. Abbildung 10.11 zeigt Netzwerktopologie, Partitionierung und den jeweiligen Zustand zu verschiedenen Zeitpunkten. Jeder Knoten A bis G stellt zum einen ein eigenes Teilnetz (1 bis 7) dar, zusätzlich sind die Knoten des linken, größeren Astes (B, D, E, F und G) als Teilnetz 8 zusammengefasst.

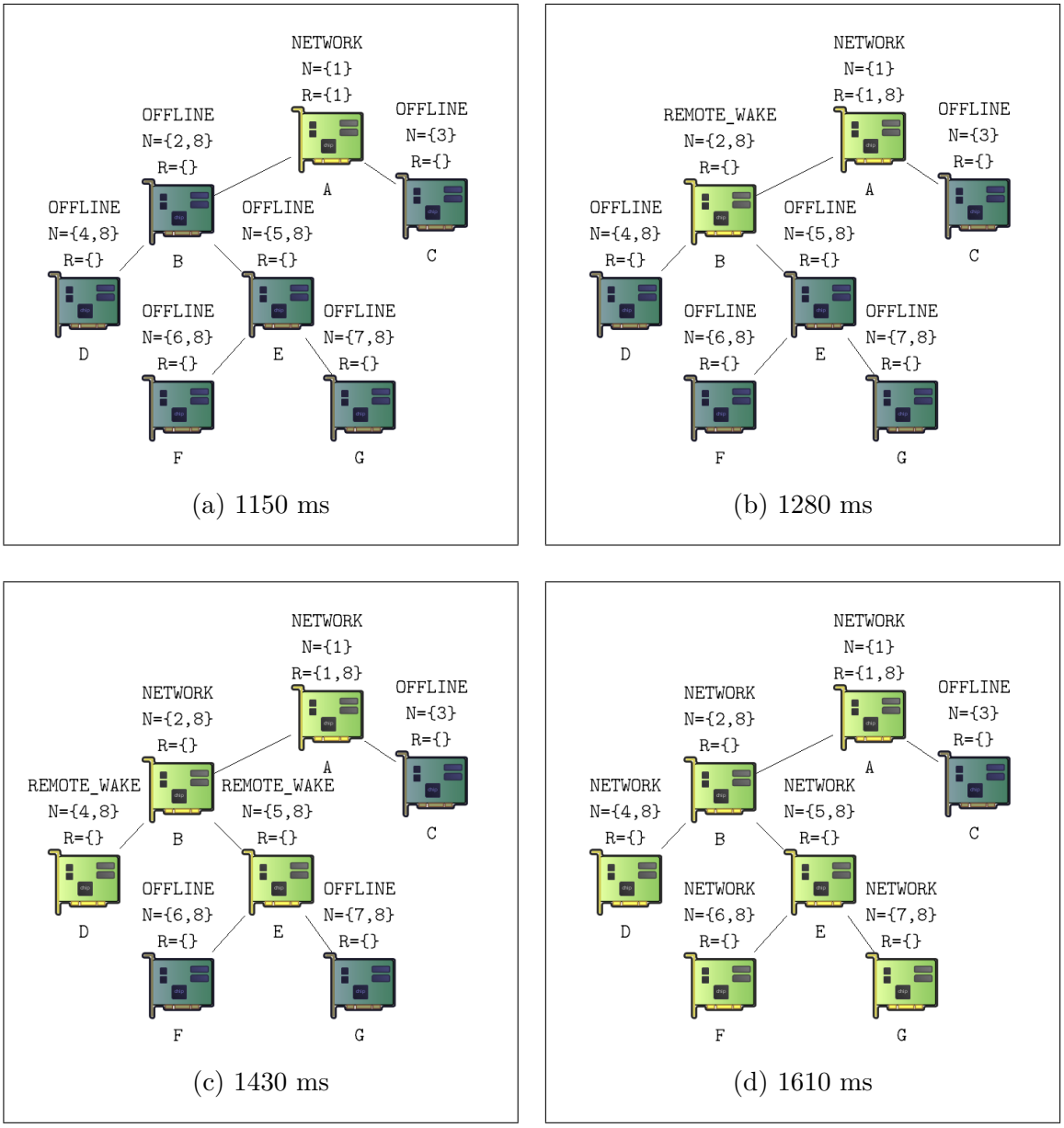


Abbildung 10.11: Fortpflanzung der Teilnetzanforderung

Zu Beginn der Simulation ist Knoten A wach und fordert nach kurzer Zeit Teilnetz 8 an. Abbildung 10.11 veranschaulicht das nachbarweise Wecken vom Moment der Anforderung bis das Teilnetz wach ist. Die letzten beiden Knoten F und G sind 480 ms nach Anforderung des Teilnetzes wach (Zustand REMOTE WAKE). Das Zustandsprotokoll kann in Anhang D.4 nachvollzogen werden.

10.5.5 Zusammenwachsen zweier Teilnetze

Ein weiteres, in der Konzeptbeschreibung von *Multinet* diskutiertes Beispiel ist das Zusammenwachsen zweier unabhängig aufgeweckter Netzbereiche (vgl. Abschnitt 8.6.1). Auch dieser Fall wurde in der Simulation nachgestellt. Abbildungen 10.12 und 10.13 zeigen die Topologie zu den wichtigsten Zeitpunkten des Ablaufs. Erneut befinden sich alle Knoten A bis G in jeweils einem eigenen Teilnetz 1 bis 8. Jeder der beiden Äste des Baumes stellt zusätzlich ein Teilnetz (Teilnetz 8 mit Knoten B, D und E, Teilnetz 9 mit Knoten C, F und G) dar. Zunächst (1150 ms, 10.12a) sind beide äußeren Knoten D und G wach und fordern jeweils das gegenüberliegende Teilnetz 9, bzw. 10 an. Beide wecken den Pfad in Richtung des jeweils angeforderten Teilnetzes auf (1290 ms, 10.12b), was bei Knoten A schließlich zusammenläuft (1430 ms, 10.13a). Das von A angeforderte, asynchrone *Keepalive* erreicht schließlich Knoten B und C, die wiederum die noch schlafenden Nachbarn in den entsprechenden Teilnetzen (E und F) wecken, so dass nun, knapp 500 ms nach Anforderung, beide Teilnetze wach und miteinander verbunden sind (1610 ms, 10.13b). Der Zustandsverlauf findet sich im Protokoll in Anhang D.5.

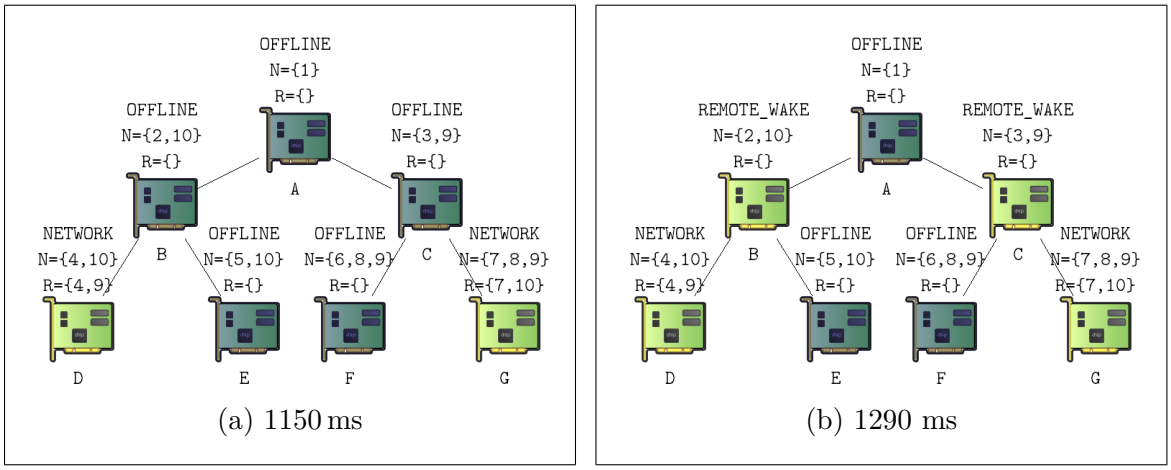


Abbildung 10.12: Anfordern der beiden gegenüberliegenden Teilnetze und Wecken des Pfades zwischen beiden Netzsegmenten

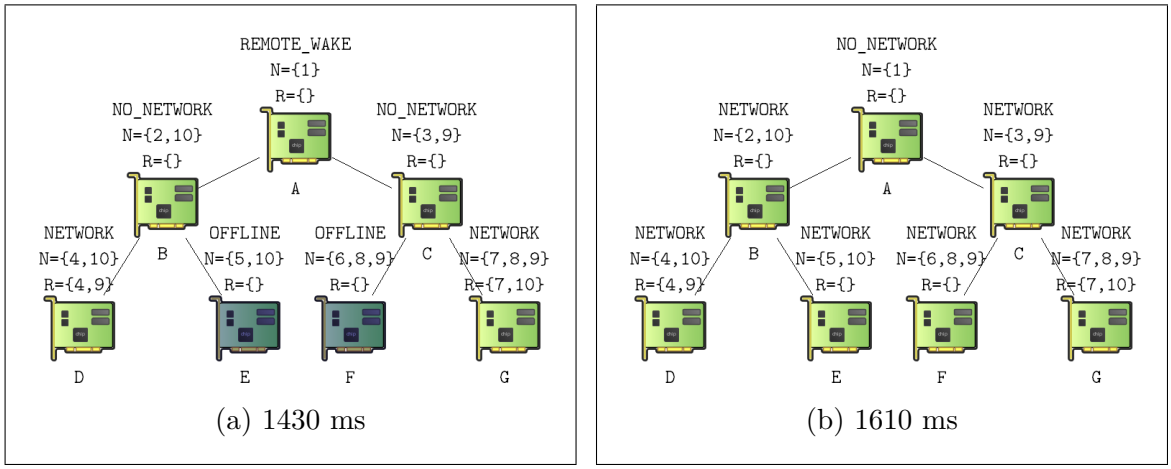


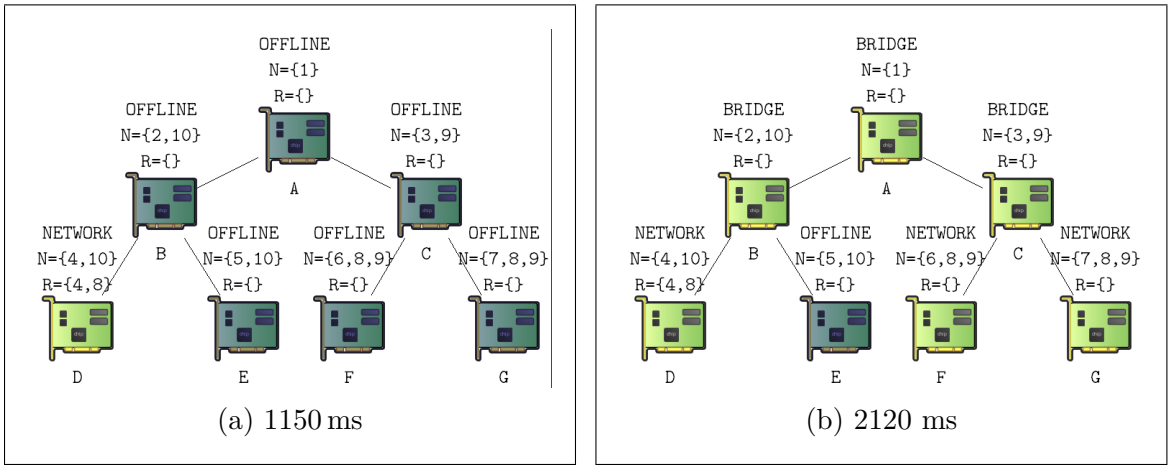
Abbildung 10.13: Zusammenwachsen der Netzbereiche und Wecken der verbleibenden Knoten

10.5.6 Anfordern eines entfernten Teilnetzes

Das Aufbauen und Halten eines Pfades zu einem entfernten Teilnetz wird in dieser Simulation nochmals verdeutlicht. Das simulierte Netzwerk ist wieder der symmetrische Baum aus dem vorherigen Beispiel (vgl. Abbildung 10.14a). Dieses Mal fordert Knoten A (zum Zeitpunkt 1150 ms) das entfernt liegende Teilnetz 8 an, welches aus den Zwei Endknoten des gegenüberliegenden Astes (F und G) besteht. Das nachbarweise Wecken pflanzt sich über Knoten B, A und C bis zum Zielteilnetz fort, zuletzt wachen Knoten F und G auf. Diese sind 580 ms nach Anforderung des Teilnetzes durch A wach (Zustand REMOTE WAKE). Die übrigen Knoten, welche nicht im angeforderten Teilnetz liegen, müssen wach bleiben, um das Netzwerk nicht zu separieren. A, B und C haben jeweils mehr als eine aktive Netzwerkverbindung und bleiben deswegen im Zustand BRIDGE (2120 ms, 10.14b) bis das Teilnetz wieder einschläft. Anhang D.6 zeigt das Protokoll der Simulation mit Zustandsverläufen aller Knoten.

10.6 Zusammenfassung

Das Ziel, die konzeptgemäße Funktionsweise der Implementierung von *Multinet* anhand verschiedener, beispielhafter Szenarien zu überprüfen und Erkenntnisse in Konzept und Implementierung rückfließen lassen zu können, wurde durch die in diesem Kapitel beschriebenen Simulationen erreicht. Dazu wurde zunächst das Simulationsframework OMNeT++ als Basis für die folgenden Tests vorgestellt, und die Einbindung der platt-



(a) 1150 ms (b) 2120 ms

Abbildung 10.14: Aufbauen eines Pfades zu einem entfernten Teilnetz

formunabhängigen *Multinet*-Implementierung in die Simulation beschrieben. Über eine Wirkkette von Auswertungsprogrammen konnten die aus der Simulation gewonnenen Ergebnisvektoren visualisiert und analysiert werden. Die konzeptgemäßen Eigenschaften von *Multinet* wurden dann über eine Reihe von Testfällen veranschaulicht und validiert. Wesentliche Funktionen, die gezeigt wurden, sind das grundlegende Anfordern, Wachhalten und Freigeben von Teilnetzen, nachbarweises Wecken, die Arbitrierung beim Wachhalten von Teilnetzen, das Brücken mehrerer, aktiver Netzsegmente und das koordinierte Einschlafen von Knoten.

11 Evaluation von Multinet im Testfahrzeug

11.1 Übersicht

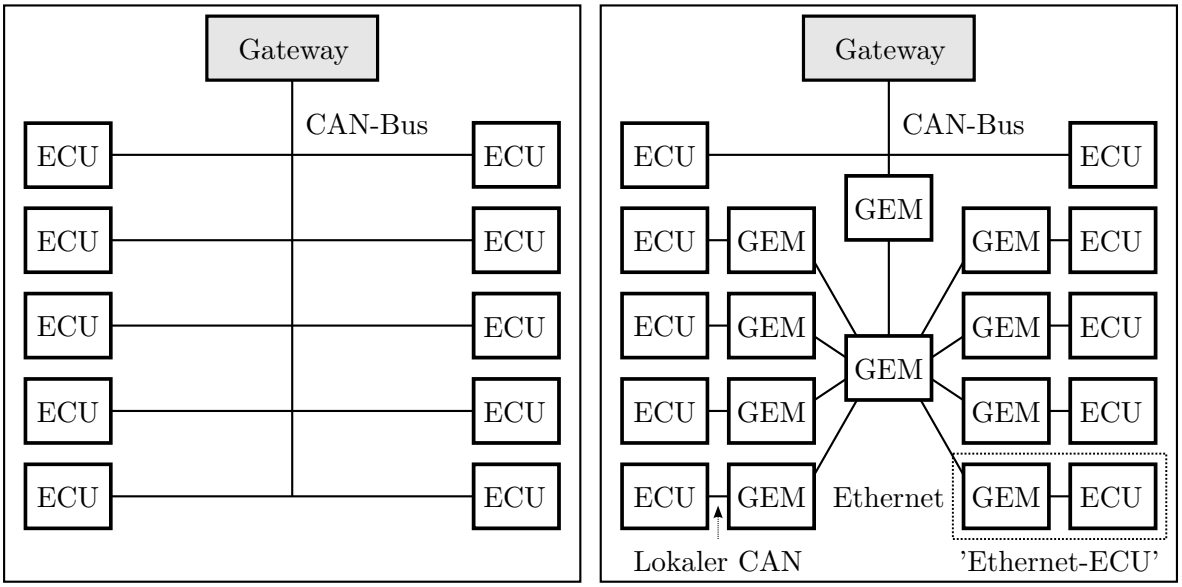
Die in den vorangegangenen Kapiteln vorgestellte *Multinet*-Architektur wurde in einem Erprobungsfahrzeug umgesetzt, um deren Funktionsfähigkeit unter realen Bedingungen zu evaluieren. Der Aufbau des Fahrzeuges erfolgte als Kooperationsprojekt zwischen Continental und Audi. Im Folgenden wird auf die Anforderungen an die Realisierung und auf den Aufbau und die Integration der Komponenten eingegangen.

11.2 Ziele und Anforderungen

Die Umsetzung von *Multinet* erfolgte im Rahmen eines Kooperationsprojektes zwischen Continental und Audi. Der Kern des Projektes war es, Ethernet-Teilnetzbetrieb mit *Multinet* in die reale Steuergerätearchitektur des Erprobungsfahrzeugs zu integrieren. Diese verwendet als Netzwerktechnologie CAN, weswegen Teile des Busses durch ein 1000BASE-T Gigabit-Ethernet Netzwerk ersetzt wurden. Der CAN-Datenverkehr der ECUs wurde mittels der für dieses Projekt entwickelten Hardwareplattform, die als CAN/Ethernet-Gateway mit *Multinet*-Implementierung diente, über UDP getunnelt. Für die Steuergeräte ist dieser Tunnel dabei vollkommen transparent. Innerhalb der so entstandenen Ethernet-Topologie konnte Teilnetzbetrieb mit *Multinet* umgesetzt werden. Entscheidungen, zu welchem Zeitpunkt welche Steuergeräte angefordert werden, wurden durch Auswertung der CAN-Daten auf den einzelnen Knoten anhand festgelegter Anforderungsbedingungen getroffen. Durch die Zusammenarbeit mit Audi konnte auf vorausgehende Arbeiten aufgesetzt werden, in deren Rahmen bereits CAN-basierter Teilnetzbetrieb in einem Erprobungsfahrzeug umgesetzt wurde.

Abbildung 11.1: Testfahrzeug Audi A8 mit Continental-Aufdruck

Audi stellte den Testträger (Audi A8 4.2 TDI) zur Verfügung (vgl. Abbildung 11.1), sowie die notwendigen Informationen über die Steuergeräte-Architektur des Fahrzeugs. Dies umfasst essentielle Informationen über den gesamten Netzwerk-Datenverkehr der Steuergeräte auf dem CAN-Bus, die Verbauorte der Steuergeräte und insbesondere die Bedingungen, wann welches Steuergerät benötigt wird und aufgeweckt werden muss. Ohne dieses Wissen wäre eine Umsetzung von Teilnetzbetrieb in einer existierenden, realen Steuergerätearchitektur nicht möglich gewesen. Bei einem derart tiefen Eingriff in die Steuergerätearchitektur des Fahrzeuges mussten Randbedingungen eingehalten werden. Ein sicherer Betrieb des Fahrzeugs musste zu jeder Zeit sichergestellt sein. Daher – und um die Zulassung im Straßenverkehr aufrecht zu erhalten – konnten keine Steuergeräte eingebunden werden, die der aktiven oder passiven Sicherheit, der Fahrdynamikregelung oder der elektronischen Steuerung des Antriebsstranges dienen. Deswegen wurde die Umsetzung auf Steuergeräte aus dem Komfort-, Body- und Instrumentenbereich beschränkt. Des Weiteren sind heutige Architekturen nicht auf das Abschalten einzelner Knoten ausgelegt. Das Ausbleiben von CAN-Botschaften der Steuergeräte würde zu Fehlerspeichereinträgen, Warnleuchten und Signaltönen, sowie dem Abwerfen einzelner Fahrzeugfunktionen führen. Deswegen musste zusätzlich eine Nachrichtensimulation implementiert werden, die – über den Zustand der *Multinet*-Teilnetze gesteuert – die zyklischen CAN-Botschaften der gerade abgeschalteten ECUs an deren Stelle in das Restfahrzeug sendet. Wird das Auto abgestellt, sieht das fahrzeuginterne Netzwerkmanagement vor, dass sich alle Steuergeräte nach einer definierten Zeit ohne Buskommunikation (Busruhe) ausschalten. Das eingebettete Ethernet-Netzwerk muss diese Busruhe insbesondere in Richtung Restfahrzeug einhalten, um die restlichen Steuergeräte nicht fälschlicherweise wach zu halten. Mit der auf diese Anforderun-



(a) Über CAN vernetzte Steuergeräte (b) Eingebettetes Ethernet-Netz

Abbildung 11.2: Integration von Ethernet in die CAN-Infrastruktur

gen hin ausgelegten Systemarchitektur konnte das vorgestellte Netzwerkmanagement- und Wake-Up-Konzept im realen Einsatz evaluiert werden. Die Ergebnisse erlauben Rückschlüsse auf zukünftiges Systemdesign sowie auf das Einsparpotential und stellen den Ausgangspunkt für Schritte in Richtung einer möglichen Standardisierung für den weitreichenden Einsatz in künftigen Fahrzeugarchitekturen dar.

11.3 Systemarchitektur des Testfahrzeugs

Bei dem Testfahrzeug handelt es sich um einen Audi A8 4.2 TDI quattro. Die umfangreiche Ausstattung an elektronischen Funktionen des Fahrzeugs bot eine gute Auswahl an Steuergeräten, die in die *Multinet*-Testarchitektur aufgenommen werden konnten, ohne die Fahrsicherheit zu gefährden.

11.3.1 Integration von Ethernet

Die Integration von Ethernet in die Architektur des Testfahrzeugs stellte eine wichtige und umfangreiche Voraussetzung für die Evaluation von *Multinet* und dem EDM dar. Hierzu wurden Teile der existierenden CAN-Infrastruktur durch ein 1000BASE-T Gigabit-Ethernet-Netzwerk ersetzt und der reale Datenverkehr über dieses getunnelt.

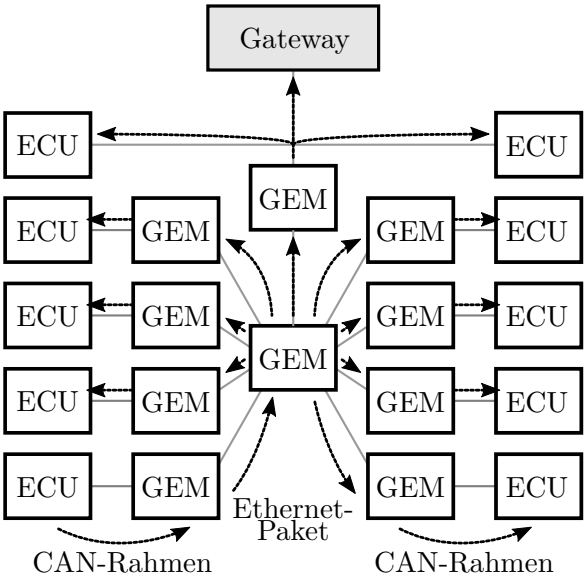


Abbildung 11.3: Tunnelung der CAN-Botschaften über UDP/IP

Zu diesem Zweck wurde das GEM (Gigabit Ethernet Modul) entwickelt (vgl. Abschnitt 11.3.2) welches die Knoten des Ethernet-Netzwerks (Endknoten und Switches) darstellt. Die GEMs ermöglichen über eine CAN-Schnittstelle die Anbindung der realen Steuergeräte und des Restfahrzeus. Dies ist in Abbildung 11.2 verdeutlicht. Jedes Steuergerät ist über einen Punkt-zu-Punkt-CAN mit einem GEM verbunden. Dieses tunnelt die CAN-Rahmen des Steuergerätes als UDP-Datagramm über IP und Ethernet. Jedes Paar aus CAN-Steuergerät und GEM verhält sich daher wie ein Ethernet-Steuergerät im Netzwerk. Jenes GEM, welches über CAN an den Restbus und an das zentrale Gateway angeschlossen ist, nimmt eine besondere Rolle ein, da es die Schnittstelle zwischen *Multinet* und dem Restfahrzeug darstellt. In Anlehnung an die Border-Router, die im Internet die Grenzrouter eines Autonomen Systems darstellen, wird es im weiteren Verlauf als Border-GEM (BGEM) bezeichnet.

CAN/UDP-Tunnel

Das Prinzip des Tunnels ist in Abbildung 11.3 dargestellt. Jedes GEM empfängt über dessen lokalen CAN-Bus die Botschaften des angeschlossenen Steuergerätes, kapselt sie jeweils in ein UDP-Datagramm und versendet dieses über das Internetprotokoll an die gerichtete Broadcast-Adresse des Subnetzes. Auf diese Weise erhalten alle anderen GEMs das Datagramm, entkapseln die darin enthaltene CAN-Botschaft wieder und stellen sie über den lokalen CAN an das angeschlossene Steuergerät zu. Durch das

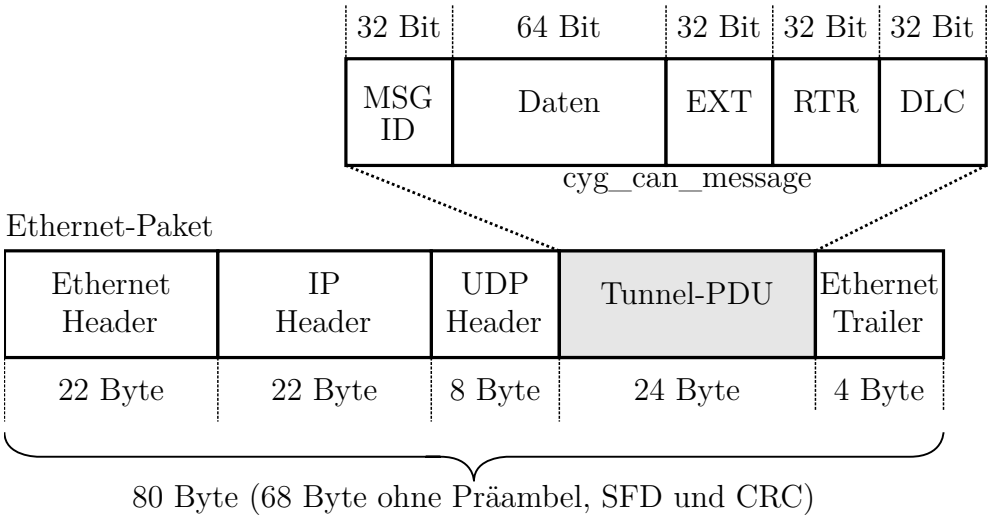


Abbildung 11.4: Datagrammaufbau der getunnelten CAN-Botschaften

BGEM werden alle CAN-Botschaften aus dem Restfahrzeug in den Tunnel gesendet, und *vice versa*. Damit ist das eingebettete Ethernet-Netzwerk für die Steuergeräte vollständig transparent. Die Kommunikation auf dem CAN-Bus ist sichergestellt und alle Fahrzeugfunktionen werden aufrechterhalten. Der Tunnel wurde im Rahmen einer durch den Autor betreuten Masterarbeit [105] auf den GEMs implementiert. Die Umsetzung erfolgte in der Programmiersprache C auf dem Betriebssystem *eCos* (Embedded Configurable Operating System) [53], welches auf den GEMs zum Einsatz kam (Abschnitt 11.3.2). *eCos* bringt einen CAN-Treiber sowie einen TCP/IP-Protokollstapel (FreeBSD) [53] mit.

Der Tunnel-Dienst besteht aus einem Sende- und einem Empfangs-Thread. Der Sende-Thread empfängt CAN-Frames unter Benutzung der CAN-Treiberschnittstelle von *eCos* und verschickt sie über einen UDP-Socket an die Broadcast-Adresse. Dabei wird die Datenstruktur `cyg_can_message`, in welcher der Treiber Botschaftskennung, Daten und Datenlänge speichert, als Nutzdaten in dem UDP-Datagramm gekapselt. Der Empfangs-Thread wartet an Port 7654/UDP blockierend auf Datagramme, entkapselt die `cyg_can_message` wieder und versendet sie über die CAN-Treiberschnittstelle. Abbildung 11.4 zeigt den Aufbau der UDP-Datagramme mit der gekapselten Struktur `cyg_can_message` als Nutzdaten. Die Größen der jeweiligen Felder der Struktur ergeben sich aus der Definition von `cyg_can_message` in *eCos* sowie Speicherlayout und Padding. Dabei ist die Struktur vier-Byte-ausgerichtet.

Listing 11.1 zeigt den Mitschnitt einer getunnelten CAN-Botschaft. Der Mitschnitt wurde mit dem Programm *Wireshark* aufgenommen. Die CAN-Nachricht selbst beginnt bei Offset 2Ah mit der Botschaftskennung 00000640h. Der DLC ist 08h und findet sich bei Offset 3Eh. Die folgenden Nullen dienen der Speicherausrichtung der gesamten Struktur.

Listing 11.1: Mitschnitt einer CAN-Botschaft im Tunnel

```
0000   ff ff ff ff ff ff 00 10   11 12 12 05 08 00 45 00
0010   00 34 a1 d2 00 00 40 11   54 8e c0 a8 01 09 c0 a8
0020   01 ff 04 00 1d e6 00 20   85 b7 00 00 06 40 a0 4d
0030   8a a6 7a 82 20 01 00 00   00 00 00 00 00 00 08 00
0040   00 00
```

Abbildung 11.5 zeigt die Funktion des Tunnels beim Einsatz im Testfahrzeug anhand der zyklischen CAN-Botschaft des Heckdeckel-Steuergerätes (HDSG). Die Botschaft enthält (unter anderem) Signale, welche die aktuelle Position des Heckdeckels (HD_Position) sowie die aktuelle Bewegungsrichtung der Aktuatorik (HD_Bewegung_auf, HD_Bewegung_zu) anzeigen. Während der Aufzeichnung wurde der Heckdeckel per Knopfdruck zunächst geschlossen und nach einer kurzen Pause

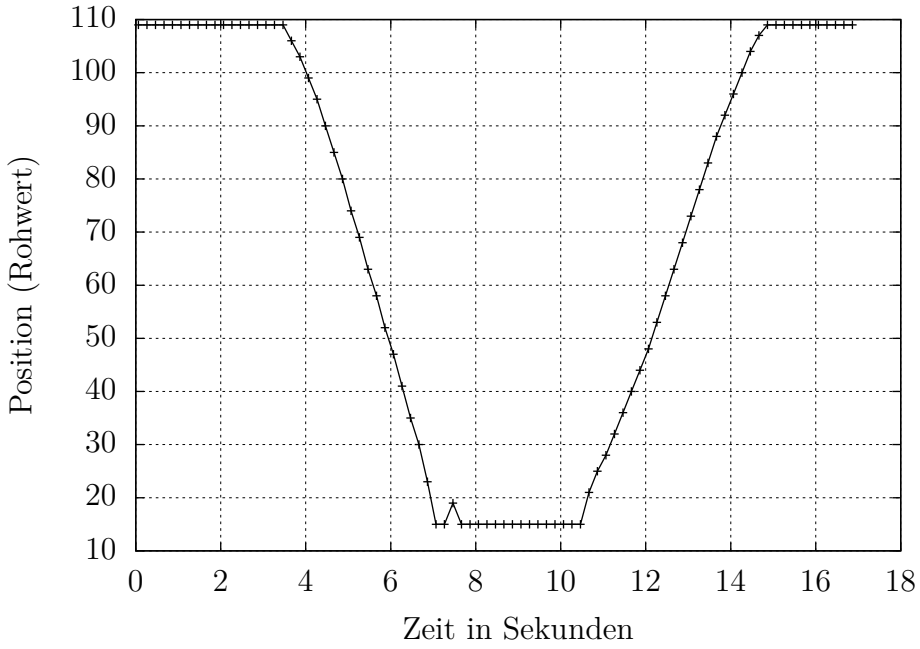


Abbildung 11.5: Zeitverlauf des Signals 'HD_Position' der getunnelten Heckdeckel-Botschaft beim Öffnen und Schließen des Heckdeckels

wieder geöffnet. Das Mitprotokollieren der getunnelten CAN-Botschaften erfolgte im Ethernet-Netzwerk mittels des Programmes *Wireshark*.

Jeder Datenrahmen des CAN-Busses wird über das eingebettete Ethernet-Netzwerk getunnelt. Die Botschaften der ausgewählten Steuergeräte werden dabei jeweils von dem GEM getunnelt, an welchem sie angeschlossen sind, während die Nachrichten von den übrigen Steuergeräten und vom zentralen Gateway durch das BGEM in das Netz weitergegeben werden. Auf dem verwendeten Bus gibt es 120 zyklische CAN-Botschaften, die mit Zykluszeiten zwischen 10 ms und 2000 ms gesendet werden. Aus der CAN-Datenbank ließ sich unter Berücksichtigung des DLC damit eine durchschnittliche Datenrate von 133 kbit/s errechnen. Dies entspricht bei einer Ethernet-Framelänge von 80 Byte (vgl. Abbildung 11.4) einer Datenrate von 816 kbit/s im Tunnel, oder einer Netzwerklast von 0,8 % bei einer verfügbaren Datenrate von 1 Gbit/s.

Teilnetzbetrieb mit *Multinet*

Das eingebettete Ethernet-Netzwerk ist transparent in das Steuergerätenetzwerk integriert und stellt eine Testplattform für den Einsatz von *Multinet* und des Energy-Detect-Moduls (Kapitel 6) dar. *Multinet* wurde hierfür auf jedem GEM implementiert und die Steuergeräte in Teilnetze partitioniert (vgl. Abschnitt 11.3.3). Auf jedem GEM befindet sich zusätzlich eine Softwarekomponente, die CAN-Botschaften auswertet und (ausgehend von den Signalwerten) Teilnetze bei *Multinet* anfordert und freigibt. Die GEMs sind mit EDMs ausgestattet und können somit von ihren Nachbarn geweckt werden. Das angeschlossene CAN-Steuergerät wird dann durch das GEM über den lokalen CAN aufgeweckt. Wacht ein Steuergerät selbst auf, beispielsweise durch Betätigung einer Taste, wird das GEM vom Steuergerät über den CAN-Bus aufgeweckt. Beim Abschalten eines Knotens wird zunächst das CAN-Steuergerät schlafen gelegt, indem Busruhe vorgetäuscht wird. Anschließend schaltet sich das GEM selbst aus. Im folgenden Abschnitt wird der Aufbau des GEMs genauer beschrieben.

11.3.2 Gigabit-Ethernet-Modul

Wie in den vorausgegangenen Abschnitten dargestellt, dient das GEM (Fotografie von Gerät und Hauptplatine in Abbildung 11.6) als Ethernet-Endknoten, Tunnel-Adapter und als Ethernet-Switch. Es bildet gleichzeitig die Schnittstelle zwischen Ethernet-Netz

(a) GEM mit Gehäuse (b) Platine des GEM

Abbildung 11.6: Gigabit-Ethernet-Modul

und der CAN-basierten Steuergerätearchitektur des Testfahrzeugs. In Abbildung 11.7 ist das Blockdiagramm des GEMs dargestellt. Die Spannungsversorgung des GEM besteht aus einem Eingangsfilter zur Unterdrückung von Störungen auf der Versorgungsleitung und wird direkt an die Autobatterie angeschlossen. Aus der Batteriespannung werden Schaltregler zur Erzeugung der Spannungen 1,2 V, 3,3 V und 5,0 V über einen Schalttransistor gespeist.

Zur Datenkommunikation stehen sechs ISO 11898-konforme CAN-Controller, ein *FlexRay*-Controller und ein 10/100 Mbit/s Ethernet-MAC zur Verfügung. Aufgrund der vielen Netzwerkschnittstellen wird der für den Einsatz im Automobil konzipierte Microcontroller häufig in Fahrzeug-Gateways verbaut. Deswegen, und insbesondere aufgrund des integrierten Ethernet-MAC, wurde er für das GEM verwendet. Der zweite, wichtige Baustein ist die integrierte Ethernet-Multiport-Bridge BCM53115M von Broadcom [18]. Die Bridge verfügt über sieben Ethernet-Ports, davon fünf mit integrierten 10/100/1000BASE-T Ethernet-PHYs. Bei den beiden anderen Ports wird die MII-Schnittstelle des Ethernet-MAC zur Anbindung eines externen PHYs oder eines Prozessors herausgeführt. Der MAC des PowerPC ist über RvMII (Reverse Media Independent Interface) mit einem der beiden MACs der Multiport-Bridge verbunden. An dem anderen MAC ist ein externer Ethernet-PHY (BCM89616) über GMII (Gigabit Media Independent Interface) angeschlossen. Zur Konfiguration und Steuerung der Bridge und der PHYs durch den PowerPC wird der Management-Bus MDIO (Management Data Input/Output) der MII-Schnittstelle verwendet. An jedem Ethernet-Port wurde ein Energy-Detect-Modul installiert, um das GEM über alle Ports aufwecken zu können (vgl. Kapitel 6). Die Wecksignale der EDMs werden nach dem Booten durch den Microcontroller ausgelesen, um den Port zu ermitteln,

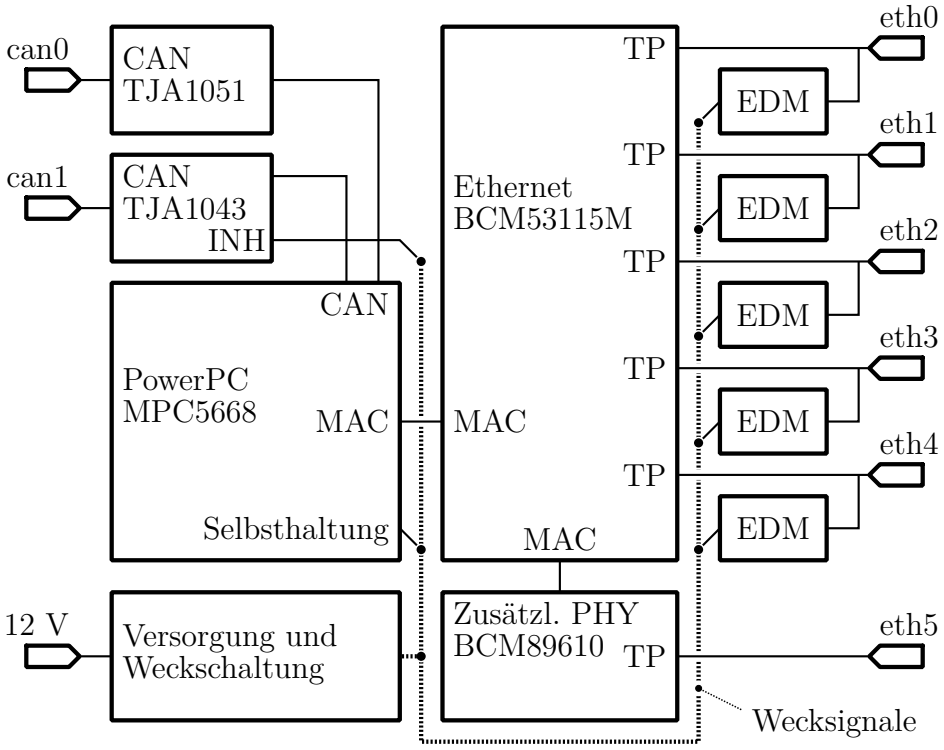


Abbildung 11.7: Blockdiagramm des Gigabit-Ethernet-Moduls

über den das GEM geweckt wurde. Zur Kommunikation über CAN wurden zwei CAN-Transceiver verbaut, die an zwei CAN-Controller des MPC5668 angeschlossen wurden. Der TJA1043 von NXP an Schnittstelle CAN1 ist weckfähig, der TJA1051 an CAN0 nicht. Beide Transceiver sind in der Lage, bis zu 1 Mbit/s zu übertragen. Damit verfügt das GEM nach außen hin über sechs weckfähige 10/100/1000BASE-T Ethernet-Ports (*eth0* - *eth5*) und zwei CAN-Schnittstellen (*can0, can1*). Da das GEM nur durch *can1* aufgeweckt werden kann, wird diese Schnittstelle zur Kommunikation mit der Steuergerätearchitektur des Testfahrzeugs eingesetzt. Die Schnittstelle *can0* wurde als optionale Möglichkeit zur Kommunikation mit Test- oder Loggingsystemen über CAN vorgesehen.

Das GEM kann über eine Reihe unterschiedlicher Weckmechanismen geweckt werden. Die Wecksignale aller weckfähigen Komponenten (EDMs und CAN-Transceiver), sowie ein Selbsthaltesignal des Microcontrollers, werden über eine ODER-Schaltung zusammengeführt und schalten die Betteriespannung über einen Transistor. Das Weckkonzept wird in Abbildung 11.8 veranschaulicht. Nach erfolgtem Wake-Up werden die einzelnen Wecksignale durch den Microcontroller ausgewertet, um die Ursache des Aufwachens festzustellen. Anschließend aktiviert der Controller die Selbsthaltung, wodurch die Spannungsversorgung unabhängig vom Zustand der Wecksignale aktiv

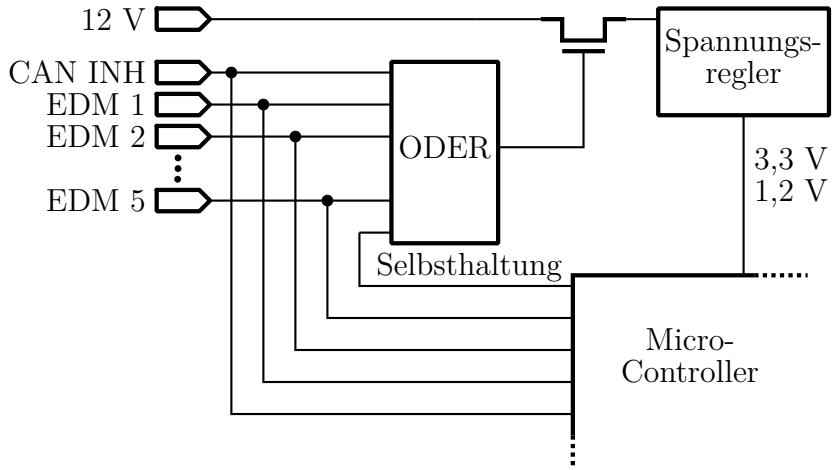


Abbildung 11.8: Weckkonzept des GEMs (Vereinfachtes Blockschaltbild)

bleibt. Die EDMs werden dann wie in Kapitel 6 beschrieben von der Leitung abgekoppelt.

Als Betriebssystem wird *eCos* (Embedded Configurable Operating System) eingesetzt. Dabei handelt es sich um ein Echtzeitbetriebssystem für eingebettete Systeme. *eCos* ist in hohem Maße konfigurierbar und so auf die jeweilige Anwendung adaptierbar. Der enthaltene TCP/IP-Protokollstapel mit POSIX-konformer (Portable Operating System Interface) Socketschnittstelle, der integrierte CAN-Treiber und die Eignung für eingebettete Systeme mit geringem Speicher waren ausschlaggebend für die Wahl von *eCos* als Betriebssystem für das GEM. Für die Zielplattform MPC5668 wurde dankenswerterweise eine Portierung durch Jochen Gerster von der Firma *Freescale* zur Verfügung gestellt. Die Integration des Betriebssystems erfolgte im Rahmen der durch den Autor betreuten Masterarbeit von Daniel Lammering [105]. In [53] werden alle wichtigen Aspekte der Softwareentwicklung unter *eCos* beschrieben.

11.3.3 Verwendete Steuergeräte, Topologie und Simulation

Für die Umsetzung im Testfahrzeug wurde eine Auswahl an Steuergeräten des Komfort-CANs getroffen. Insgesamt ergibt sich damit eine Gesamtzahl von 12 vernetzten Steuergeräten. Das System wurde mit der in Kapitel 10 vorgestellten Umgebung in OMNeT++ modelliert und simuliert. Das BGEM verbindet das Netzwerk mit dem Restfahrzeug. Heckdeckelsteuergerät (HSG) und Türsteuergeräte (TL, TR) sind direkt am BGEM angeschlossen, sowie die Botschaftssimulation (SIM) und Switch 1, an den

Tabelle 11.1: Auswahl an Steuergeräten für die Umsetzung von *Multinet*

Steuergerät	Beschreibung
Türsteuergeräte	Zuständig für Entriegelung, Fensterheber, Spiegelverstellung, Blinker, und weitere Türfunktionen. Rechte und linke Seite.
Anhängersteuergerät	Anhängerlebenderkennung, Ein- und Ausfahren der Anhängerkupplung.
Heckdeckelsteuergerät	Steuert das Öffnen und Schließen der automatischen Heckklappe.
Sitzsteuergeräte	Sitzverstellung, Heizung und Sitzmemory für alle vier Sitze. Ein Steuergerät pro Sitz.
Ventilblöcke	Massagefunktion und Lordosenverstellung durch aufblasbare Polster. Ein Steuergerät pro Sitz.

wiederum die Sitzsteuergeräte (SVL, SVR, SHL, SHR) angebunden sind. Am Botschaftssimulator sind das Anhängersteuergerät (ASG) und Switch 2 mit den Ventilsteuergeräten (VVL, VVR, VHL, VHR) angeschlossen. Abbildung 11.9 zeigt die Topologie und Partinitionierung. In der dargestellten Simulation fordert das BGEM Teilnetz 5 an, also die vier Sitzsteuergeräte. Switch 1 ist im Zustand BRIDGE.

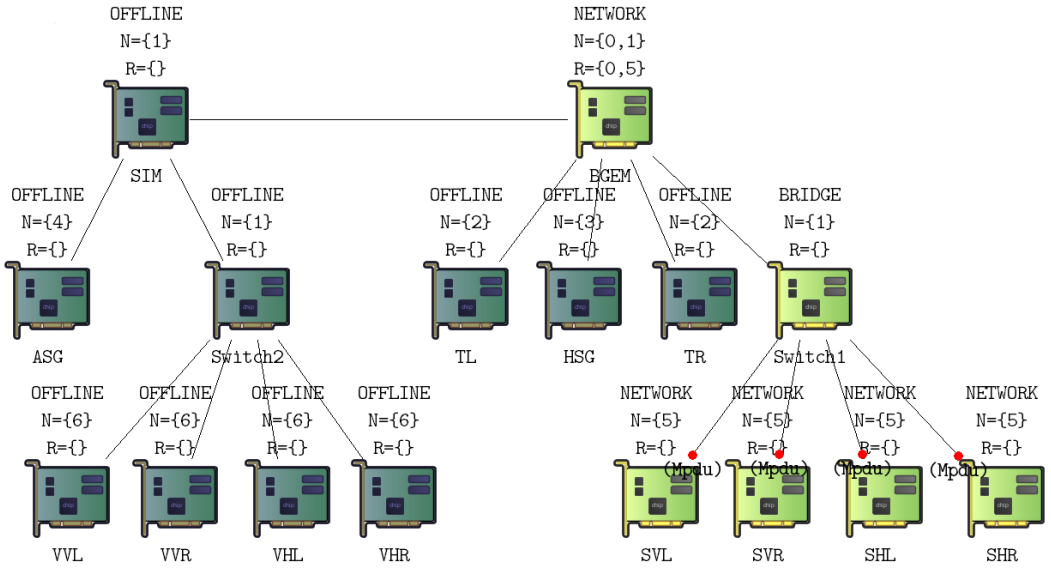


Abbildung 11.9: Simulation der Anforderung der Sitzsteuergeräte durch das BGEM

11.3.4 Mechanischer Aufbau

Durch diese Topologie ergibt sich eine Gesamtzahl von 16 GEMs für den Aufbau. Aus Gründen der Zugänglichkeit für Messungen, Entwicklungsarbeit und Fehlersuche wurden alle GEMs in einem 4x4-Rack im Kofferraum des Fahrzeuges integriert (vgl. Abbildung 11.10). Die CAN-Leitungen der ausgewählten Steuergeräte wurden an zugänglichen Stellen aufgetrennt, und beide Enden in den Kofferraum verlängert. Über einen Brückenstecker konnte somit das Steuergerät bei Bedarf wieder an den CAN-Bus angeschlossen werden. Ohne Brückenstecker wurden die CAN-Leitungen an das jeweilige GEM angeschlossen und waren nicht mehr mit dem Bus verbunden. Im Rack wurden für jedes GEM entsprechende Versorgungsleitungen (Batterie und Masse) verlegt, die in einem Stecker zusammengeführt und über eine Sicherung und einen Schalter an die Fahrzeugbatterie angeschlossen wurden. Über den Schalter konnten die GEMs abgeschaltet (oder rückgesetzt) werden.

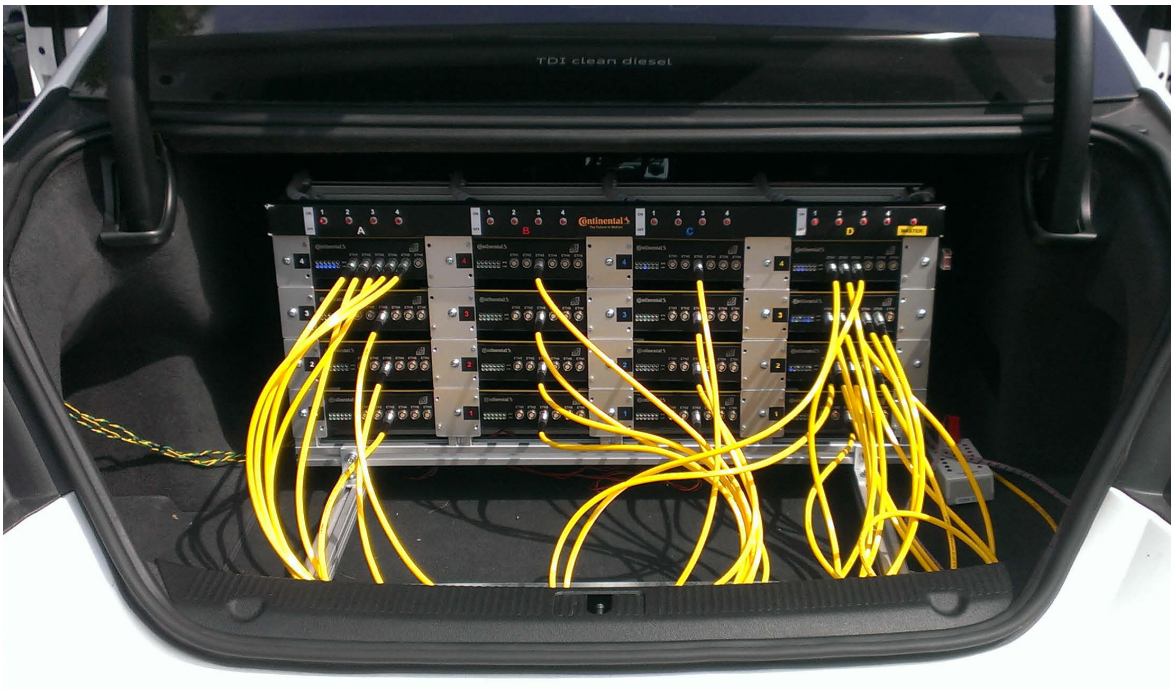


Abbildung 11.10: Rackaufbau im Kofferraum des Testfahrzeugs

11.4 Softwarearchitektur und Integration von Multinet

Die Integration von *Multinet* in das Ethernet-Netzwerk des Testfahrzeugs erforderte die Einbettung in eine geeignete Softwarearchitektur.

- Client-Softwarekomponente, die anhand von Anforderungsbedingungen Teilnetze anfordert.
- Softwarekomponente, um das lokal angeschlossene CAN-Steuergerät aufzuwecken und schlafenzulegen, sowie um am CAN-Netzwerkmanagement des Gesamtfahrzeugs teilnehmen zu können.
- Botschaftssimulation, um die Botschaften gerade schlafender Steuergeräte für die übrigen Busteilnehmer zu simulieren.

Die Integration der aufgelisteten Module wird in den folgenden Abschnitten dargestellt und kurz beschrieben.

11.4.1 Client-Software und Anforderungsbedingungen

In einer Fahrzeugarchitektur, in der *Multinet* auf realen Steuergeräten läuft, würden die Softwarekomponenten der Steuergeräte-Software Teilnetze über die Dienstschnittstelle anfordern. Da dies im Testfahrzeug nicht der Fall ist, wurde eine Anwendung auf den GEMs integriert, die diese Funktion übernimmt. Jedes GEM verfügt über eine eigene Clientanwendung (*Multinet*-Client, MCLI). Das grundlegende Konzept hierbei ist, dass jeder Client die CAN-Botschaften des an das jeweilige GEM angeschlossenen Steuergerätes auswertet und abhängig vom Inhalt der darin enthaltenen Signale, Teilnetze anfordert, so lange die jeweilige Anforderungsbedingung erfüllt ist. Das BGEM als Schnittstelle zum Restfahrzeug interpretiert alle Nachrichten vom Restbus des Fahrzeugs. Die Festlegung von Anforderungsbedingungen erfordert genaue Kenntnis der Steuergerätearchitektur, der gesamten Buskommunikation und der funktionalen Zusammenhänge zwischen den einzelnen Steuergeräten, über die in der Regel nur der Fahrzeughersteller verfügt. Die Ergebnisse vorhergehender Arbeiten von J. Meyer *et al.* [27] konnten hierfür im Rahmen des Kooperationsprojektes zwischen Continental und Audi herangezogen werden. Die erarbeiteten Anforderungsbedingungen für die ausgewählten Steuergeräte des als Testfahrzeug verwendeten Audi A8 wurden als Grundlage für die Evaluierung von *Multinet* zur Verfügung gestellt.

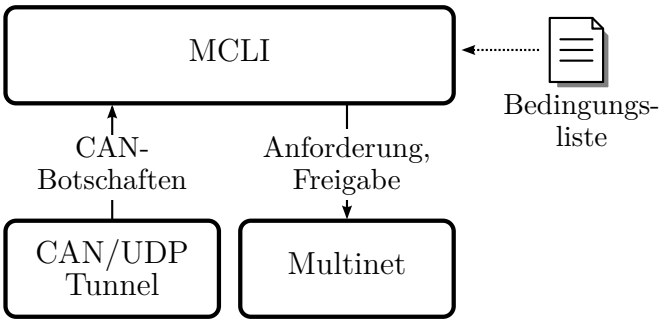


Abbildung 11.11: *Multinet*-Client

Jede Anforderungsbedingung stellt dabei einen Zustand oder Zustandswechsel eines Signals innerhalb einer CAN-Botschaft dar (Tabelle 11.2). Die aktuelle Geschwindigkeit des Fahrzeugs werde also beispielsweise als Rohwertsignal in einer CAN-Botschaft versendet und es gelte die Bedingung 'Rohwert des Geschwindigkeitssignals *kleiner* als 1000' (Anm.: Vorfaktor 0,01 s). Dann ist die Bedingung erfüllt, sobald die Geschwindigkeit des Fahrzeugs 10 km/h unterschreitet. Auf diese Weise ist es möglich, alle durch CAN-Signale repräsentierten Fahrzeug- und Umgebungszustände als Anforderungsbedingungen zu verwenden. Eine detaillierte Auflistung der jeweiligen Anforderungsbedingungen und zugehörigen Signalwerte im vorliegenden Dokument ist aus Gründen der Geheimhaltung nicht möglich. An dieser Stelle wird daher nur auf das zugrundeliegende Konzept eingegangen, und die expliziten Anforderungsbedingungen werden als gegeben vorausgesetzt. Der MCLI auf jedem GEM wertet die CAN-Botschaften aus, die vom lokalen CAN-Steuergerät (beim BGEM vom Restfahrzeug) kommen, und vergleicht sie mit einer Liste an Anforderungsbedingungen. Dabei gibt es für jedes

Tabelle 11.2: Typen von Anforderungsbedingungen

Bedingungstyp	Beschreibung
Steigende Flanke	Signalbit wechselt von '0' auf '1'.
Fallende Flanke	Signalbit wechselt von '1' auf '0'.
Flankenwechsel	Signalbit ändert sich im Vergleich zur vorhergehenden Botschaft.
Wert gleich	Der Rohwert des Signals entspricht einem festgelegtem Wert.
Wert ungleich	Der Rohwert des Signals weicht vom festgelegtem Wert ab.
Wert größer	Der Rohwert des Signals ist größer als der festgelegte Wert.
Wert kleiner	Der Rohwert des Signals ist kleiner als der festgelegte Wert.
Wert wechselt	Der Rohwert des Signals ändert sich im Vergleich zur vorhergehenden Botschaft.

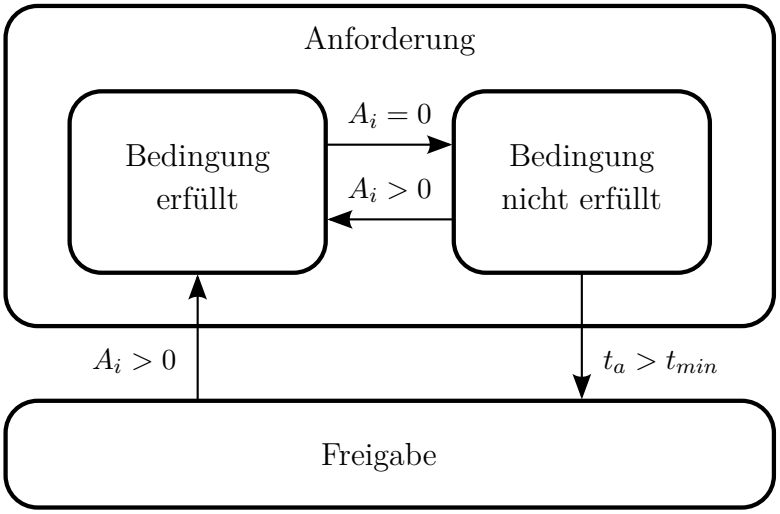


Abbildung 11.12: Automatendarstellung der Anforderungslogik für eine Gruppe i

Teilnetz eine bool'sche Kombination erfüllter Bedingungen, die zur Anforderung der Gruppe führt. MCLI greift dazu auf die Dienstschnittstelle von *Multinet* zu (Abbildung 11.11). Die einzelnen Bedingungen können als bool'sche Werte c_n aufgefasst werden, die WAHR sind, solange die Bedingung erfüllt ist, andernfalls als FALSCH. Bedingt beispielsweise c_1, dass der Rohwert des Geschwindigkeitsignals größer 1000 ist, dann wäre c_1 WAHR sobald das Fahrzeug schneller als 10 km/h fährt. Über bool'sche Funktionen $A_i = F(c_1, ..., c_n)$ können die elementaren Bedingungen für jedes Teilnetz i miteinander kombiniert werden. Ist das Ergebnis A_i WAHR, wird Teilnetz i angefordert. Im einfachsten Falle stellt $A_i = F(c_1, ..., c_n)$ eine ODER-Verknüpfung aller für das Teilnetz relevanten Bedingungen dar. Jedoch sind auch komplexere Kombinationen möglich, wie etwa die gleichzeitige Erfüllung verschiedener Konditionen. Als Beispiel hierfür sei eine Signalflanke genannt, die nur bei abgeschalteter Zündung zur Anforderung eines Teilnetzes führen soll. In diesem Fall wird die Flankenbedingung mit einer Gleichheitsbedingung des Zündungssignals aus der Klemmenstatus-Botschaft UND-verknüpft.

Die Anforderungslogik kann also für jedes Teilnetz durch einen endlichen Automaten dargestellt werden (Abbildung 11.12). Ein Teilnetz i wird von MCLI angefordert, sobald dessen Aktivierungsfunktion A_i als WAHR ausgewertet wird. Es bleibt dann angefordert, so lange A_i WAHR ist, wenigstens jedoch für eine Mindestanforderungsdauer t_{min}. Erlischt die Bedingung und A_i wird FALSCH, wird das Teilnetz wieder freigegeben, wenn die Zeitdauer der Anforderung t_a die Mindestanforderungszeit t_{min}

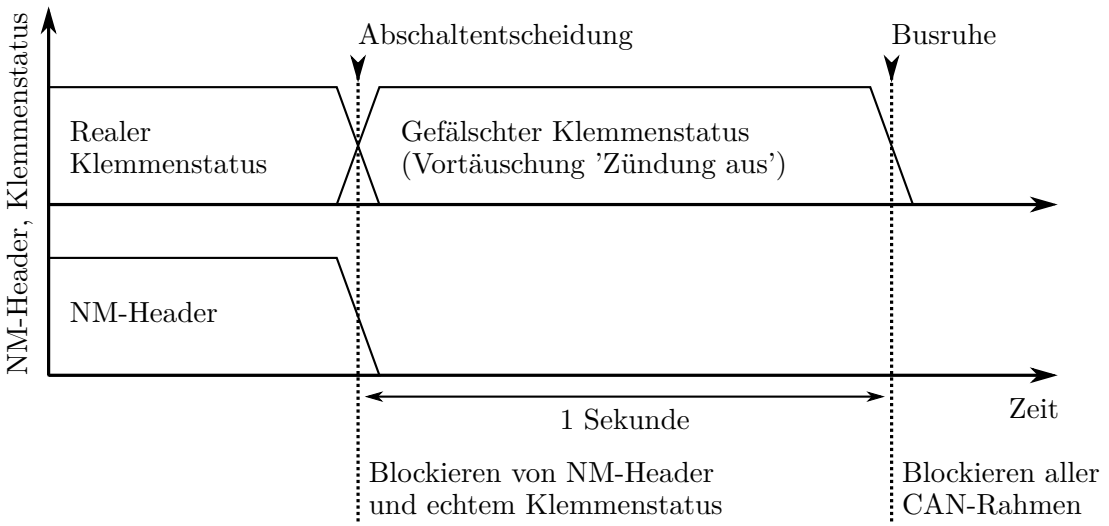


Abbildung 11.13: Sequenz zur Vortäuschung von Busruhe auf lokalem CAN

überschritten hat. Durch die Mindestanforderungsdauer ist es möglich, auch singuläre Ereignisse wie z. B. Flankenwechsel oder Wertänderungen von Signalen als Anforderungsbedingung zu nutzen, bei denen die Bedingung nur für einen sehr kurzen Zeitraum als WAHR ausgewertet wird. Zusätzlich bleibt hierdurch das System stabil bei häufig wechselnden Zuständen der Anforderungsbedingungen und durchläuft nicht zyklisch Aufwach- und Abschaltphasen.

11.4.2 Wecken und Abschalten der CAN-Steuergeräte

Beim Aufwecken oder Abschalten eines *Multinet*-Knotens muss das lokal angeschlossene CAN-Steuergerät zunächst ebenfalls geweckt werden bzw. schlafen gehen. Da Steuergeräte bei Aktivität auf dem CAN-Bus aufwachen, erfolgt das Wecken automatisch, sobald das GEM mit dem Weiterleiten der CAN-Botschaften aus dem Tunnel beginnt. Das Herunterfahren erfordert eine Sequenz an Botschaften, die dem Steuergerät das Schlafengehen des Busses beim Abstellen des Fahrzeugs vortäuscht, so dass sich dieses selbst auch schlafen legt. Die Steuergeräte des Busses bereiten sich auf das Abschalten vor, sobald die Zündung abgestellt wird. Der Zustand der Zündung (Klemme 15) wird auf dem Bus in einer 'Klemmenstatus'-Botschaft signalisiert. Beim CAN-Netzwerkmanagement (vgl. Abschnitt 4.3) sendet jedes Steuergerät, welches noch nicht bereit ist um schlafen zu gehen, zyklisch Netzwerkmanagement-Header (NM-Header).

Wurden für den Zeitraum von einer Sekunde keine NM-Header mehr empfangen, müssen sich alle Busteilnehmer abschalten. Durch Blocken der NM-Header aus dem Netzwerk und durch Fälschen der 'Klemmenstatus'-Botschaft kann also dem lokalen Steuergerät eine bevorstehende Busruhe vorgetäuscht werden, und dieses schaltet sich aus. Dieser Ablauf ist in Abbildung 11.13 dargestellt. Die Abschaltentscheidung erfolgt zum Zeitpunkt, zu dem *Multinet* aus dem Zustand *STANDALONE* in den Zustand *SHUTDOWN* wechselt, also wenn der Knoten nicht mehr angefordert wird. Falls das CAN-Steuergerät selbst noch NM-Header sendet, signalisiert es damit, dass es sich noch nicht abschalten kann. In diesem Falle muss das Zeitfenster von einer Sekunde bis zur Busruhe mit jedem empfangenen NM-Header hinausgezögert werden, bis das Steuergerät das Senden einstellt. Falls der *Multinet*-Knoten vor Ablauf der Sequenz wieder angefordert wird (z. B. durch Empfang einer *Keepalive*-Botschaft oder durch MCLI), muss die Sequenz abgebrochen werden und die Blockade der NM-Header und der realen Klemmenstatus-Botschaft wieder aufgehoben werden. Dadurch wird verhindert, dass das CAN-Steuergerät schlafen geht.

11.4.3 Botschaftssimulation

Da das Abschalten von ECUs in der existierenden Steuergerätearchitektur nicht vorgesehen ist, führt dieses zu Fehlermeldungen und Funktionsabwürfen. Der Grund hierfür ist das Ausbleiben der zyklischen CAN-Botschaften der ausgeschalteten Steuergeräte. Andere ECUs erwarten diese Botschaften, und melden Timeout-Fehler oder deaktivieren Funktionen sobald sie nicht mehr auf dem Bus empfangen werden. Dies kann durch eine intelligente Botschaftssimulation vermieden werden.
Ein dedizierter Knoten im Netzwerk, der in *Multinet* eingebunden ist, sendet die CAN-Botschaften der schlafenden Steuergeräte an ihrer statt in den Tunnel. Dies ist möglich, da CAN Botschaften adressiert, und keine Sender oder Empfänger. Jede CAN-Botschaft hat eine eindeutige Botschaftskennung (*Message-ID*). Es ist dabei für die anderen Busteilnehmer nicht ersichtlich, welcher Sender die Botschaft auf den Bus gesendet hat. Durch die Kommunikationsdatenbank (*dbc-File*) des Testfahrzeugs sind alle Botschaften sowie deren Absender und Zykluszeiten bekannt. Aus diesen Informationen und der Gruppierung der Steuergeräte in Teilnetze wurde eine Matrix erstellt, welche die CAN-Botschaften der in *Multinet* eingebundenen Steuergeräte mit denjenigen Teilnetzen verknüpft, in denen das jeweils sendende Steuergerät enthalten ist. Durch *Multinet* weiß die Botschaftssimulation zu jeder Zeit, welche Teilnetze gerade aktiv sind, und welche nicht. Ist keines der mit einer Botschaft verknüpften Teilnetze

aktiv, wird diese durch die Botschaftssimulation periodisch mit der zugehörigen Zykluszeit ins Netz geschickt. Sobald eines der entsprechenden Teilnetze angefordert wird, muss die Botschaftssimulation das Senden der simulierten Nachricht einstellen, da dann der echte Sender geweckt wird und die Botschaft wieder selbst verschickt. Der Inhalt der simulierten Botschaft entspricht dabei dem der zuletzt vom Steuergerät gesendeten, echten Botschaft. Dies impliziert, dass die Botschaftssimulation alle relevanten CAN-Botschaften zwischenspeichern und aktualisieren muss. Der Grund hierfür ist, dass unplausible Wertesprünge durch die simulierten Botschaften zu Fehlverhalten führen können. Zusätzlich besitzen manche CAN-Botschaften einen Botschaftszähler und eine Prüfsumme. Beides muss beim Versenden der simulierten Nachrichten berücksichtigt werden. Der Botschaftszähler muss weitergezählt werden, und die Prüfsumme muss gemäß Spezifikation berechnet und mitgesendet werden. Ein nicht mehr mitlaufender Botschaftszähler oder eine falsche Prüfsumme würden ebenfalls in Fehlermeldungen und der Deaktivierung von Funktionen resultieren.

11.4.4 Integration ins Netzwerkmanagement des Fahrzeugs

Der Zustand aller Steuergeräte eines Fahrzeugs wird auch heute über Netzwerkmanagement gesteuert. Der Unterschied zu Teilnetzbetrieb ist, dass dies busweit und nicht selektiv geschieht. Wie schon in Abschnitt 4.3 beschrieben, sendet jedes Steuergerät periodische Netzwerkmanagement-Botschaften, so lange es den CAN-Bus wachhalten möchte. Wenn für den Zeitraum von einer Sekunde keine NM-Botschaft mehr auf dem Bus gesendet wird, gehen alle Steuergeräte schlafen, und die Kommunikation auf dem Bus wird eingestellt (Busruhe). Dies geschieht beispielsweise beim Abstellen des Fahrzeugs, um alle Steuergeräte herunterzufahren und das Auto „auszuschalten“. Das eingebettete Netzwerk muss am Netzwerkmanagement des Fahrzeugs teilnehmen und die Busruhe ebenfalls einhalten. Andernfalls würden zwei Probleme auftreten: Getunnelte oder simulierte CAN-Botschaften, die trotz Busruhe ins Fahrzeug gesendet würden, hielten alle Steuergeräte wach. Die Batterie wäre dann in kurzer Zeit leer. Außerdem bliebe der Kern des Netzes (BGEM, Switches) wach und verbraucht Strom. Auch wenn dessen Leistungsaufnahme im Vergleich zur Gesamtheit aller Steuergeräte niedrig ist, würde dieser Ruhestrom dennoch die Batterie auf längere Sicht während der Standzeit leeren. Das BGEM muss daher das Ausbleiben von NM-Botschaften und die damit angeforderte Busruhe erkennen und darauf reagieren. Hierzu wurde auf dem BGEM eine weitere Softwarekomponente integriert, die das Auftreten von NM-Botschaften auf dem Bus überwacht. Mit jeder eintreffenden NM-Botschaft wird ein

Timer rückgesetzt. Nach einer Sekunde läuft dieser ab und zeigt einen NM-Timeout an. Ab diesem Zeitpunkt wird jegliches Tunneln von Botschaften ins Restfahrzeug gestoppt. Damit ist sichergestellt, dass die simulierten Botschaften, die weiterhin von der Botschaftssimulation gesendet werden, nicht ins Restfahrzeug gelangen und dieses wachhalten. Um das BGEM und den Botschaftssimulator selbst mit dem Fahrzeug abzuschalten, wird es (bzw. das Teilnetz 0, in sich beide Knoten befinden) durch die NM-Applikation angefordert, so lange kein NM-Timeout auftritt. Geht das Fahrzeug schlafen, wird Teilnetz 0 ebenfalls nicht mehr angefordert, und BGEM und Simulator schalten sich aus. Wacht das Auto wieder auf, wachen BGEM und Simulation ebenfalls wieder auf. Dabei spielt es keine Rolle, ob das Netzwerk durch das Restfahrzeug (BGEM) oder durch ein Steuergerät im *Multinet*-Netzwerk geweckt wird. In beiden Fällen wird das zugehörige GEM lokal aufgeweckt und fordert daraufhin eines oder mehrere Teilnetze an. Die durch das jeweilige Steuergerät auf dem CAN versendeten (und getunnelten) NM-Header sorgen dafür, dass auch der Rest des Fahrzeugs aufwacht.

11.5 Datenaufzeichnung

Zur Aufnahme aller relevanten Daten des Fahrzeugs und des Netzwerkmanagements wurde der gesamte Datenverkehr im Ethernet-Netzwerk mitgeschnitten. Dies wurde durch einen Laptop-PC mit dem Programm Wireshark durchgeführt. Wireshark wurde auch bereits bei der Validierung des CAN/UDP-Tunnels verwendet und ist in der Lage alle Datenpakete aufzuzeichnen, die über ein Netzwerkinterface gesendet oder empfangen werden. Sowohl getunnelte CAN-Botschaften als auch die PDUs von *Multinet* werden an die Broadcast-Adresse 192.168.1.255 gesendet und werden deswegen auch durch den Laptop-PC empfangen. Da alle CAN-Botschaften des Busses über Ethernet getunnelt werden, konnten alle relevanten Fahrzeugdaten wie beispielsweise die Geschwindigkeit, betätigte Taster, oder allgemein als Anforderungsbedingung definierte Signale mit Zeitstempel aufgezeichnet werden. Zusammen mit den Botschaften (*Keepalive*, *Query*) ließ sich der Ablauf der Anforderung, des Aufweckens, der Freigabe und schließlich des Schlafengehens der einzelnen Knoten im Zusammenhang betrachten und validieren. Um besser nachvollziehen zu können, wann welches Steuergerät wach ist, wurde auf den GEMs das Versenden periodischer Heartbeat-Pakete (vgl. Anhang B.3) implementiert. Daraus ließ sich neben der Validierung der korrekten Funktionalität auch der Verlauf der Leistungsaufnahme aller Knoten während der Testfahrten

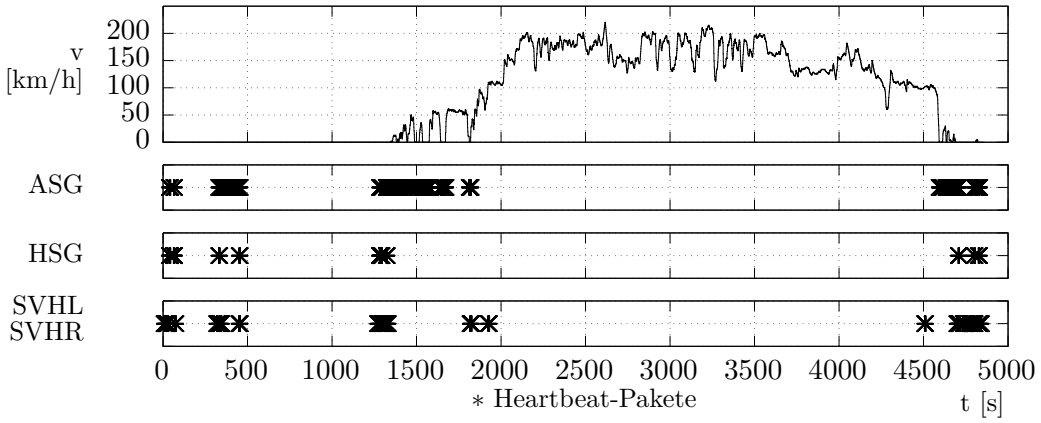


Abbildung 11.14: Aufzeichnung auf einer Testfahrt von Regensburg nach Fürth

hochrechnen und mit der Leistungsaufnahme der ausgewählten Steuergeräte ohne die Verwendung von Teilnetzbetrieb vergleichen.

11.6 Messfahrt von Regensburg nach Fürth

Auf einer Fahrt von Regensburg nach Fürth mit dem Testträger wurde der Netzwerkverkehr mitgeschnitten um den Betrieb des *Multinet*-Netzwerks zu protokollieren. Abbildung 11.14 zeigt die Heartbeat-Nachrichten des Anhängersteuergeräts, des Heckdeckelsteuergerätes und der vier Sitzsteuergeräte über den gesamten Fahrtverlauf[1]. Zur Referenz ist im obersten Diagramm die aufgezeichnete Geschwindigkeit dargestellt. Bei Ein- oder Abschalten der Zündung wird kurzzeitig das gesamte Netzwerk angefordert, dies ist für alle Teilnetze eine der Anforderungsbedingungen. Zu Beginn der Messung stand das Fahrzeug während der ersten knapp 25 Minuten geparkt, zwei Mal mit aktivierter Zündung (direkt zu Beginn, und im Bereich von ca. 300 s bis 500 s). Im zweiten Zeitintervall wurde dabei der Heckdeckel geöffnet und geschlossen. Eine entsprechende Ausschnittsvergrößerung ist in Abbildung 11.15 dargestellt. Sie zeigt die Heartbeat-Nachrichten des Heckdeckel-Steuergeräts, während es aktiv ist, also während der Kofferraumdeckel geöffnet, bzw. geschlossen wird. Dazwischen schläft es kurz und legt sich nach Schließen des Kofferraums auch wieder schlafen. Zu erkennen sind auch die zwei Sekunden, die der Knoten bis zum Ablaufen der TTL noch wach bleibt. Nach ca. 1250 s wurde die Testfahrt begonnen. Abbildung 11.16 zeigt

[1] Bei Durchführung dieser Fahrt waren die Türsteuergeräte und die Ventilsteuergeräte noch nicht im System integriert

einen zeitlichen Ausschnitt während der Fahrt durch das Stadtgebiet von Regensburg. Eine der Anforderungsbedingungen für das Teilnetz des Anhängersteuergerätes ist eine Fahrzeuggeschwindigkeit von unter zehn Kilometern pro Stunde. Dies ist erforderlich, damit das Steuergerät bei niedrigen Geschwindigkeiten eine Anhängerlebenderkennung durchführen kann. Dies ist auch im dargestellten Ausschnitt zu sehen. Sinkt die Geschwindigkeit unter zehn Stundenkilometer, wacht das ASG auf. Knapp 1800 s nach Beginn der Aufzeichnung wurde zwei Mal der Sitz verstellt, was zum Aufwachen des Sitz-Teilnetzes geführt hat. Während der Autobahnfahrt waren die Steuergeräte des *Multinet*-Netzwerks die ganze Zeit über inaktiv. Gegen Ende der Fahrt wurde noch einmal ein Sitz verstellt und auf dem Parkplatz nochmals der Kofferraum geöffnet und wieder geschlossen.

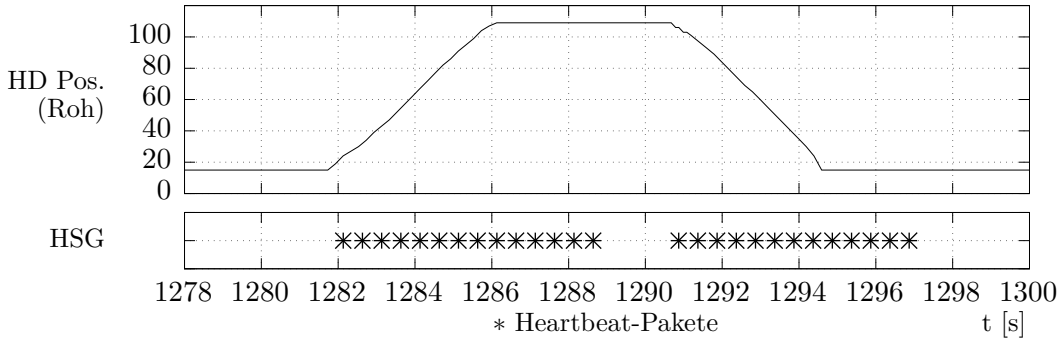


Abbildung 11.15: Öffnen und Schließen des Kofferraums vor der Fahrt

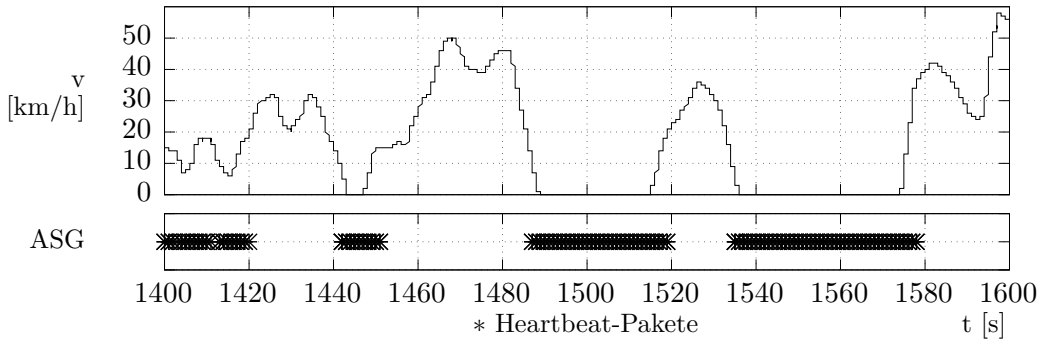


Abbildung 11.16: Ausschnitt im Stadtgebiet von Regensburg

11.7 Vorstellung des Fahrzeugs

Das Testfahrzeug wurde als erstes Auto mit Ethernet-Teilnetzbetrieb überhaupt zu verschiedenen Anlässen vorgestellt. Unter anderem bei Continental, auf mehreren internen Veranstaltungen (vgl. Abbildung 11.17), Fachtagungen und Messen, sowie öffentlich bei der Fachtagung Hanser Automotive Networks 2015 in Stuttgart und bei Nacht.Schafft.Wissen 2015 an der OTH Regensburg. Das Projekt und die zugrundeliegenden Technologien wurden in [90] öffentlich vorgestellt.

Abbildung 11.17: Das Testfahrzeug bei einer internen Veranstaltung auf dem Systemprüfkurs von Continental. Vorne im Bild der Autor (links) und sein Kollege Daniel Lammering (mitte)

12 Zusammenfassung und Ausblick

Im Rahmen der vorliegenden Arbeit wurde die Energieeffizienz der Bordnetzarchitektur heutiger Fahrzeuge aus dem Blickwinkel der Steuergerätevernetzung betrachtet. Mit der wachsenden Anzahl elektronischer Funktionen und dem damit einhergehenden Anstieg des Bedarfs an elektrischer Energie im Fahrzeug gewinnt auch die energieeffiziente Auslegung der Bordelektronik immer mehr an Bedeutung. Schon heute verursachen elektronische Komponenten (ohne Berücksichtigung von Aktoren oder Leistungselektrik) eine Leistungsaufnahme von bis zu 600 Watt, gleichbedeutend mit etwa einem Liter Mehrverbrauch pro hundert Kilometer, oder 21 Gramm CO_2 an Emissionen pro Kilometer. Um dieser Tendenz entgegenzuwirken, gibt es zahlreiche Möglichkeiten, die Bordelektronik effizienter zu gestalten. Eine davon ist Teilnetzbetrieb, das An- und Abschalten vernetzter Steuergeräte nach Bedarf. Dies wird in heute in Serie befindlichen Fahrzeugen noch nicht praktiziert. Statt nur diejenigen Komponenten zu betreiben, die gerade benötigt werden, bleiben alle Steuergeräte (teils über hundert bei Fahrzeugen der Oberklasse) während des Betriebs des Fahrzeuges eingeschaltet. Erst jetzt, mit dem steigenden Bestreben, Energie zu sparen, entstehen entsprechende technische Lösungen um Teilnetzbetrieb umzusetzen. Für den im Fahrzeug seit Jahrzehnten eingesetzten CAN-Bus gibt es bereits eine Umsetzung, die die technischen Voraussetzung für Teilnetzbetrieb erfüllt und CAN-Steuergeräte über den Bus selektiv weckfähig macht. Das durch AUTOSAR standardisierte Netzwerkmanagement wurde entsprechend erweitert, um den jeweiligen Zustand individueller Teilnetze unabhängig voneinander zu koordinieren. Für die nächste Generation der Fahrzeugvernetzung mittels Ethernet gibt es jedoch noch kein geschlossenes Konzept zur Realisierung von Teilnetzbetrieb.

An dieser Stelle setzt die vorliegende Arbeit mit einem vollständigen Konzept zur Optimierung der Energieeffizienz eines auf Ethernet basierten Steuergerätenetzwerks an. Aufbauend auf theoretischen Vorüberlegungen liefert sie konkrete Lösungsansätze zum Eneriesparen mittels Teilabschaltung von Steuergeräten, bis hin zu deren Realisierung in einem Testfahrzeug.

Der Leistungsbedarf von Ethernet selbst wurde analysiert und in Relation zu anderen Netzwerktechnologien gesetzt. Dabei wurden auch der Einfluss und die Wirksamkeit von *Energy-Efficient-Ethernet* (EEE) evaluiert, einer Erweiterung des Ethernet-Standards zur adaptiven Reduzierung des Stromverbrauchs von Netzwerk-Transceivern. Die hierfür entwickelten Modelle wurden in [50] weiterverwendet. Nicht mehr Bestandteil der vorliegenden Arbeit ist eine Betrachtung des Einflusses von EEE auf Netzwerkstandards zur Verbesserung der Dienstgüte im Fahrzeugbordnetzwerk, wie Zeitsynchronisation, Verkehrsformung und Zeitsteuerung.

Ein ruhestromloses, selektives Weckverfahren für Ethernet-Steuergeräte wurde entwickelt. Dabei wird ein Steuergerät durch die Erkennung von Link-Aktivität auf der Netzwerkleitung geweckt. Die dazu verwendete Schaltung wurde vorgestellt, simuliert und analysiert. Das Schaltungs- und Funktionsprinzip wurde patentiert. Mit dem vorgeschlagenen Verfahren ist nachbarweises, ruhestromloses Wecken von Ethernet-Steuergeräten möglich. Durch diesen Baustein ist eine der Grundvoraussetzungen für Teilnetzbetrieb bei Ethernet erfüllt.

Es wurden zwei Netzwerkmanagement-Konzepte zur softwareseitigen Realisierung von Teilnetzbetrieb entwickelt. Ein zentrales NM-Konzept, welches sich des *Simple Network Management Protocol* bedient, wurde umgesetzt und diskutiert. Anschließend wurde *Multinet*, ein umfassendes, verteiltes Netzwerkmanagement zur Zustandskoordination einzelner Knoten und Teilnetze vorgestellt und diskutiert. *Multinet* bietet Softwarekomponenten einen Dienst, um Teilnetze anzufordern und wieder freizugeben und setzt dies durch ein geeignetes Protokoll und das Wecken entsprechender Nachbarn um. *Multinet* wurde prototypisch implementiert und durch Simulation verschiedener Situationen validiert. Das Konzept ist so gestaltet, dass es zum AUTOSAR-Netzwerkmanagement kompatibel ist und in den Standard integriert werden kann.

Zuletzt wurde die Implementierung von *Multinet* im Zusammenspiel mit einer diskreten Realisierung des *Energy-Detect-Moduls* auf einem prototypischen Steuergerätenetzwerk in einem Testfahrzeug umgesetzt und unter realen Bedingungen erprobt.

Insgesamt wurde mit dieser Arbeit das Spektrum an Lösungen zur Verbesserung der Energieeffizienz der elektrisch-elektronischen Fahrzeugarchitektur auf den Bereich der Ethernet-Vernetzung ausgedehnt. Die Fahrzeughersteller sind es letztendlich, die sich aus diesem Spektrum einsetzbarer Technologien bedienen können, um den Energie-

bedarf ihrer Steuergerätenetze mit vergleichsweise geringem Aufwand zu optimieren. Wenngleich einzelne Maßnahmen, wie beispielsweise EEE, nur einen geringen Beitrag zu leisten vermögen, so steckt doch in der Summe aller Optimierungen – und damit in der lückenlosen Abdeckung aller Energieeinsparungsbereiche – ein nicht zu vernachlässigendes Potential.

Anhang

A Simulationsdaten des Energy Detect Moduls

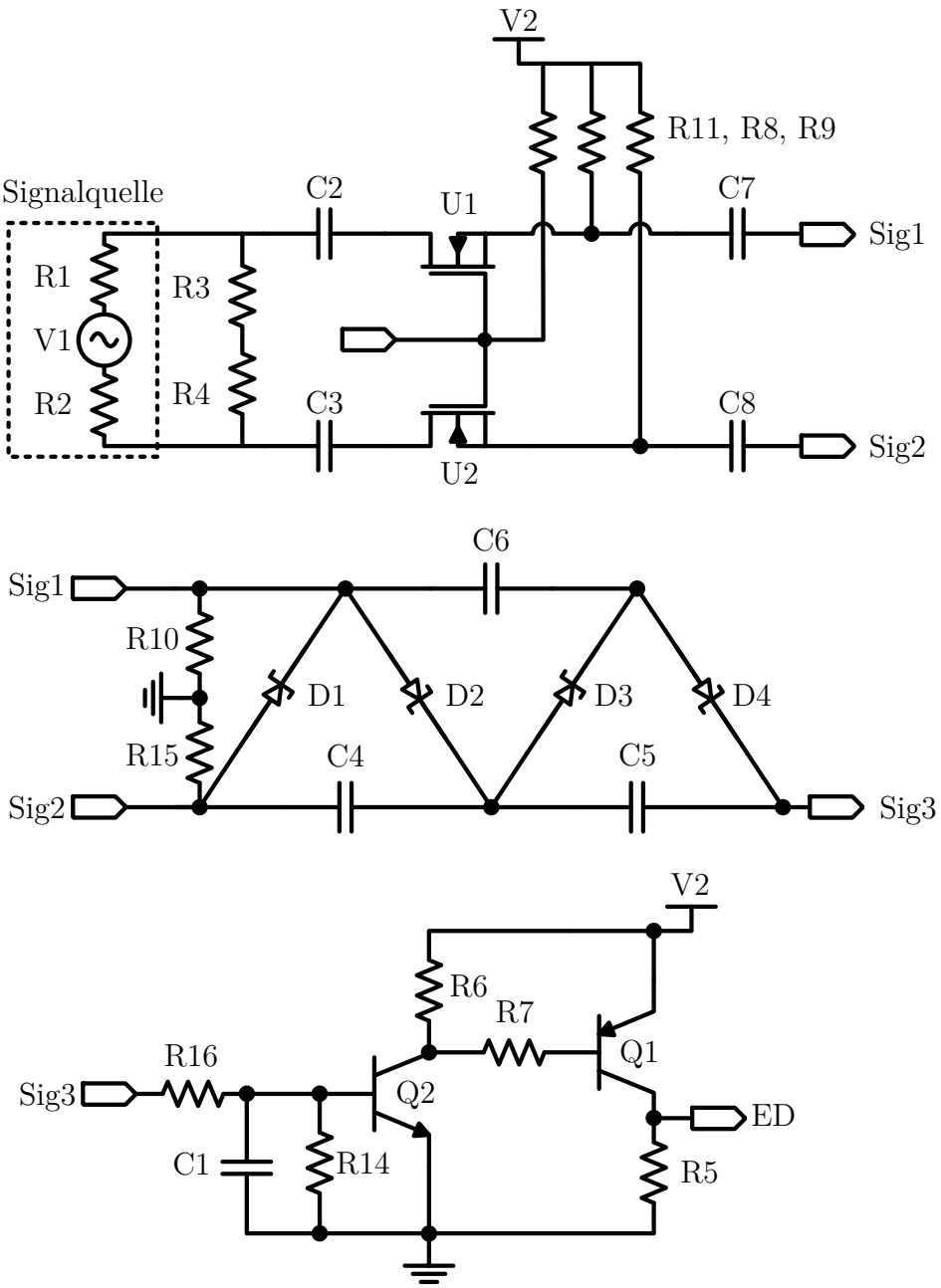
A.1 SPICE-Netzliste

Listing A.1: SPICE-Netzliste des simulierten EDM

```
V1 N004 N006 SINE(0 1856m 33Meg 0 0 0)
V2 kl30 0 12
Q1 p7 N001 kl30 0 BC857C
R5 p7 0 10k
R6 kl30 s1 10k
R7 N001 s1 1k
Q2 s1 base 0 0 BC847C
C2 N002 tp 100n
C3 N008 tn 100n
D1 gcn gp BAT54
C4 gcn gn 10n
D2 gp gn BAT54
D3 gn N003 BAT54
D4 N003 gcp BAT54
C5 gn gcp 10n
C6 gp N003 10n
C7 gp trdp_sw 10n
C8 gcn trdn_sw 10n
XU1 N002 P001 trdp_sw BSS139_L1
XU2 N008 N007 trdn_sw BSS139_L1
R8 kl30 trdp_sw 10k
R9 kl30 trdn_sw 10k
```

```
R11 kl30 N005 10k
R12 P001 N005 1k
R13 N005 N007 1k
C1 base 0 10n
R14 base 0 1Meg
R10 gp 0 10k
R15 0 gcn 10k
R1 tp N004 50
R2 N006 tn 50
R3 tp 0 50
R4 0 tn 50
R16 base gcp 1k
.model D D
.model NPN NPN
.model PNP PNP
.tran 100u
.model SW SW(Ron=0.01 Roff=1Meg Vt=3
  Vh=0.001 Lser=10n Vser=.006)
.lib small_signal.lib
.backanno
.end
```

A.2 Simulierter Schaltplan



B Definitionen der Multinet-Nachrichten

Sowohl *Keepalive*- als auch *Query*-Botschaften werden als Broadcast versendet und können als Nutzdaten über ein geeignetes Transportprotokoll (z. B. UDP) übertragen werden. Für alle Nachrichtentypen wird die gleiche Protokolldateneinheit (MPDU) verwendet, die wie folgt aufgebaut ist:

```
| TYP (16 Byte) | Param. A (16 B) | Param. B (16 B) | Param. C (16 B) |
```

In Tabelle B.1 sind die Botschaften *Keepalive*, *Query* und *Heartbeat* definiert. Der Aufbau der MPDU ermöglicht ein Hinzufügen weiterer Typen.

Tabelle B.1: Aufbau der Query-MPDU

Typ	Parameter A	Parameter B	Parameter C
10 (KEEPALIVE)	Teilnetzkennung	Priorität	Wachhaltegrund
20 (QUERY)	0	0	0
205 (HEARTBEAT)	0	0	Zustand

B.1 Query

Die *Query*-Botschaft wird beim Start eines Knotens gesendet um zu andere Knoten zum Versenden einer asynchronen *Keepalive*-Botschaft aufzufordern. Parameter werden nicht verwendet.

B.2 Keepalive

Die *Keepalive*-Botschaft wird von Knoten versendet, die ein Teilnetz aktiv wachhalten. Parameter A enthält die Kennung des entsprechenden Teilnetzes und Parameter B enthält die Prioritätszahl zur Arbitrierung. Als Parameter C kann optional ein Wachhaltegrund mitgesendet werden. Dieser wird von *Multinet* nicht ausgewertet, kann aber an die Applikationsschicht weitergegeben werden und ermöglicht eine Diagnose eines *Multinet*-Netzes durch Mitschnitt der Pakete.

B.3 Heartbeat

Die *Heartbeat*-Nachricht wurde speziell zur Auswertung von Messungen für die Evaluation von *Multinet* im Testfahrzeug hinzugefügt. Wenn *Multinet* so konfiguriert ist, dass die Nachricht verwendet wird, wird sie von jedem wachen Knoten zyklisch versendet um zu signalisieren dass der Knoten aktiv ist. Als Parameter C kann optional der Zustand des *Multinet*Kerns mitgesendet werden.

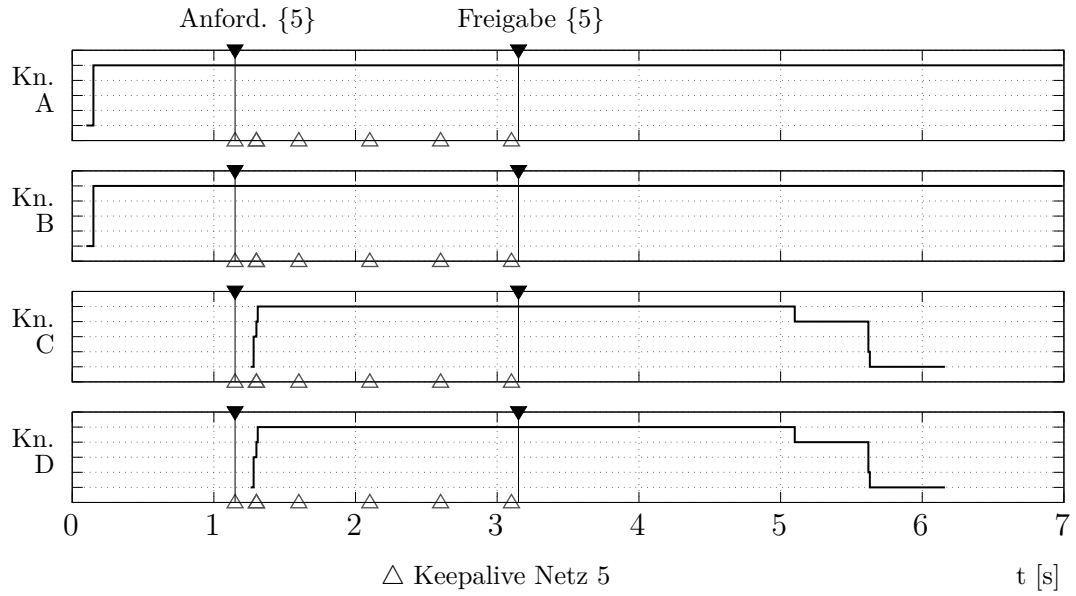
C Parametrierung der Multinet-Software

Tabelle C.1: Parameter der Multinet-Implementierung

Parameter	Beschreibung	Wert
`BASE_TIMER_MS`	Zykluszeit des Basistimers	10 ms
`T_KA_MULT`	*Keepalive*-Zykluszeit in Vielfachen des Basistimers	50 (500 ms)
`TTL_MULT`	TTL in Vielfachen der *Keepalive*-Zykluszeit	4 (2000 ms)
`HANDOVER_MULT`	Erkennungsfenster in Vielfachen der *Keepalive*-Zykluszeit	2 (1000 ms)
`T_SLEEP_MULT`	Timerwert T_SLEEP in Vielfachen des Basistimers	50 (500 ms)

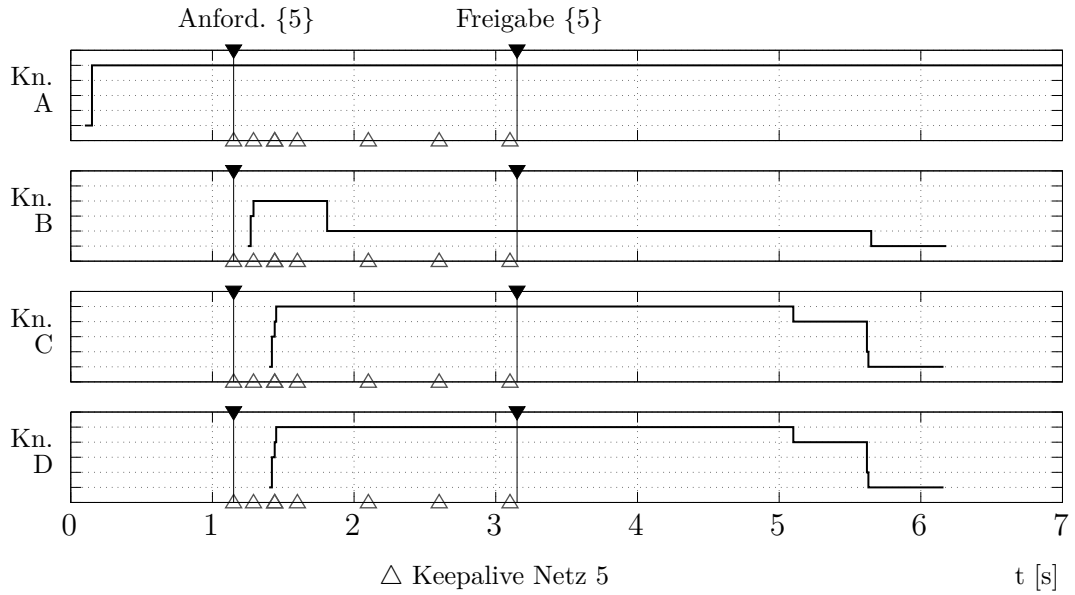
D Simulationsprotokolle

D.1 Wecken und Freigeben eines Teilnetzes



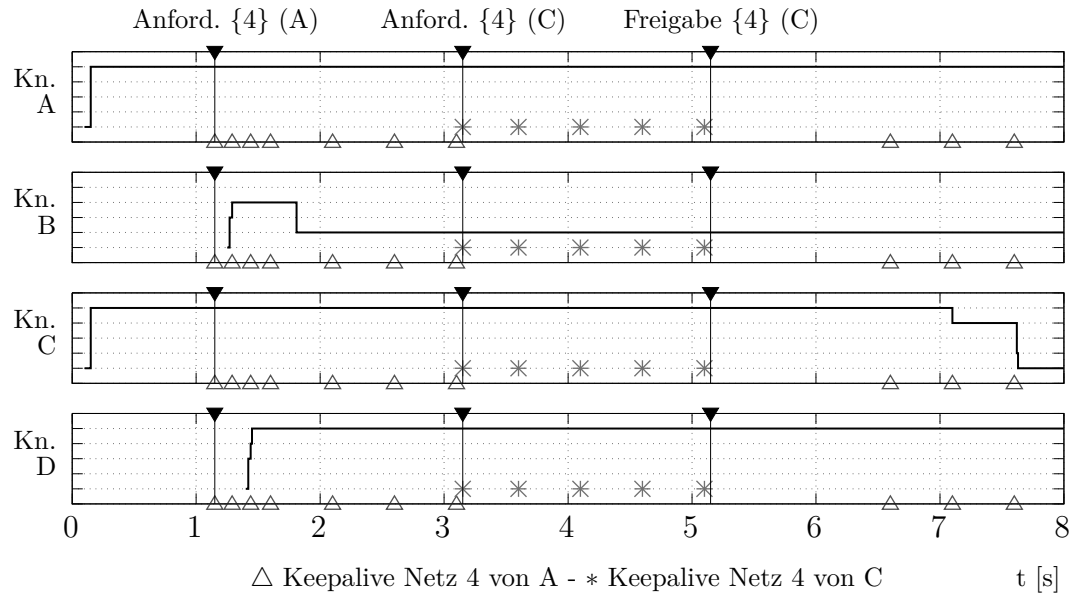
Anmerkung: Die Knotenzustände auf der Ordinate sind OFFLINE (1), BRIDGE (2), REMOTE WAKE (3), NO NETWORK (4), NETWORK (5)

D.2 Wecken eines Teilnetzes bei schlafendem Verzweigungsknoten



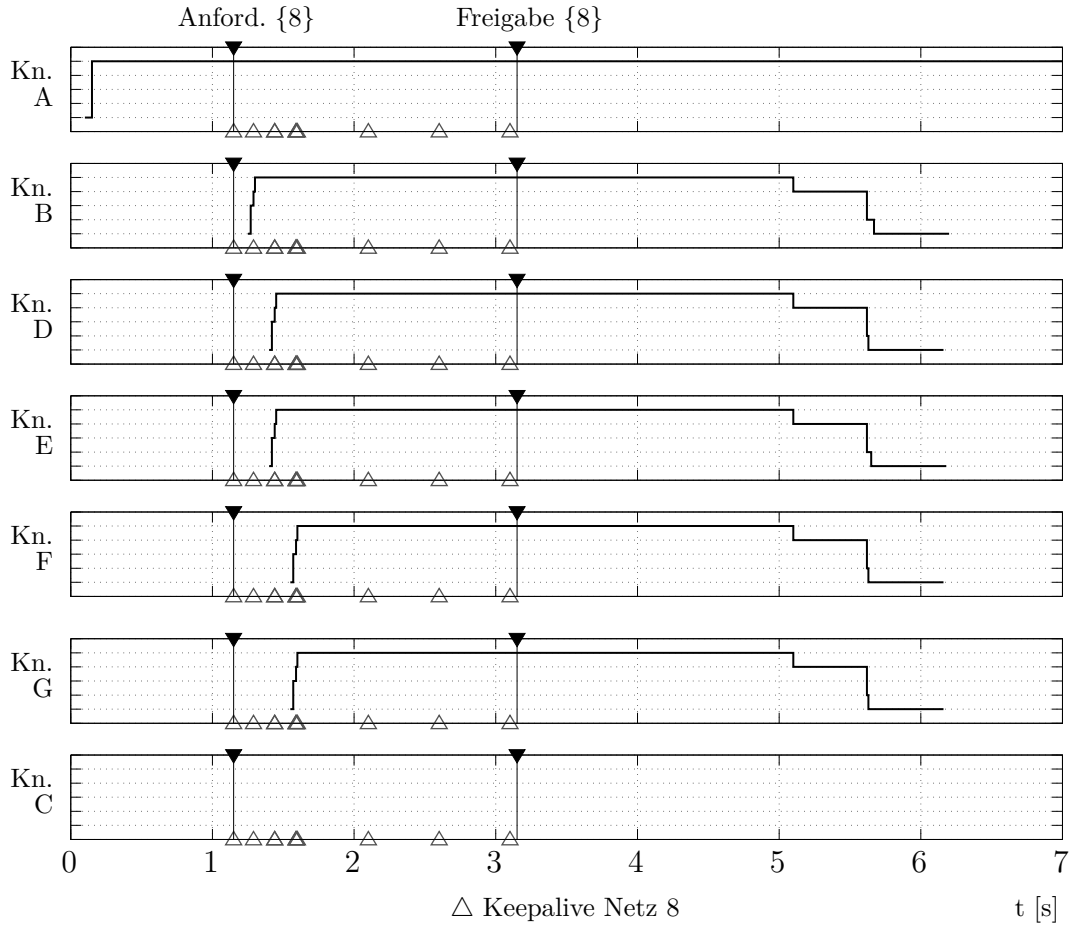
Anmerkung: Die Knotenzustände auf der Ordinate sind OFFLINE (1), BRIDGE (2), REMOTE WAKE (3), NO NETWORK (4), NETWORK (5)

D.3 Wachhalten durch mehrere Knoten gleichzeitig



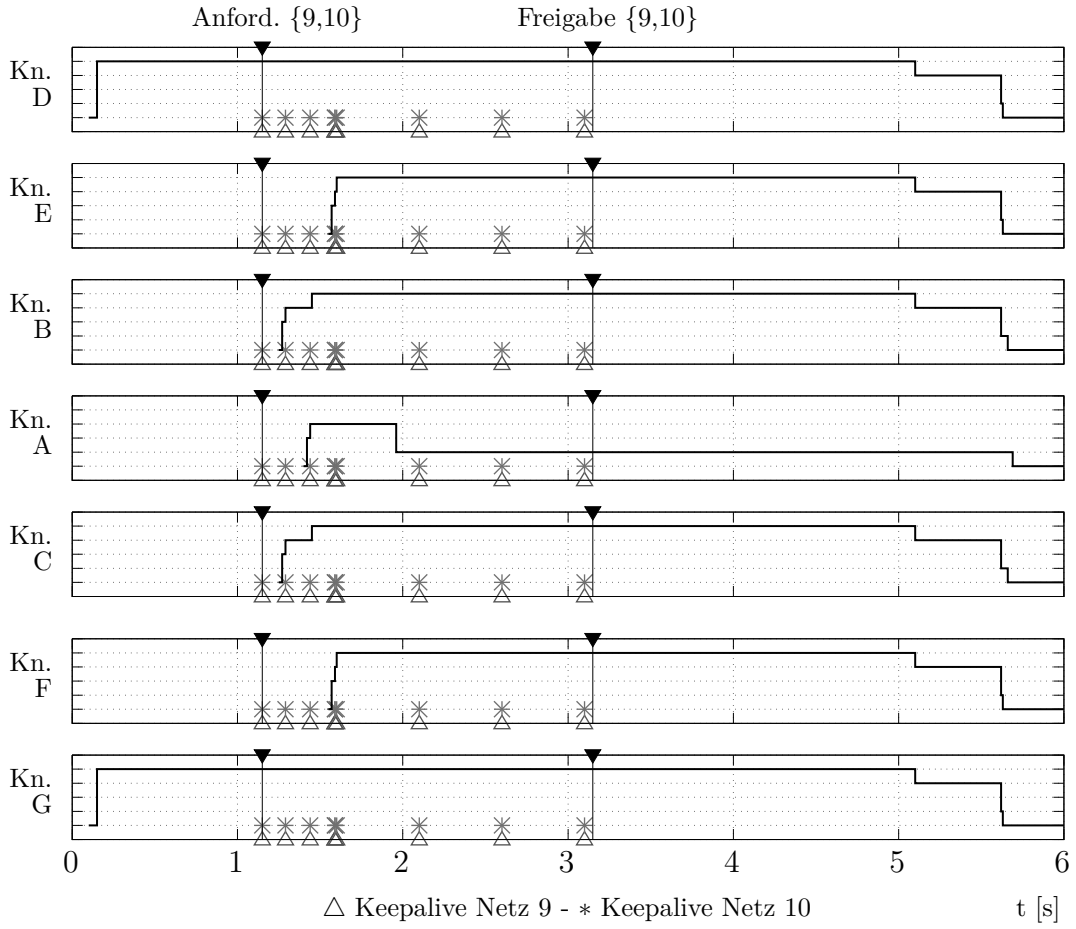
Anmerkung: Die Knotenzustände auf der Ordinate sind OFFLINE (1), BRIDGE (2), REMOTE WAKE (3), NO NETWORK (4), NETWORK (5)

D.4 Wecken eines größeren Teilnetzes



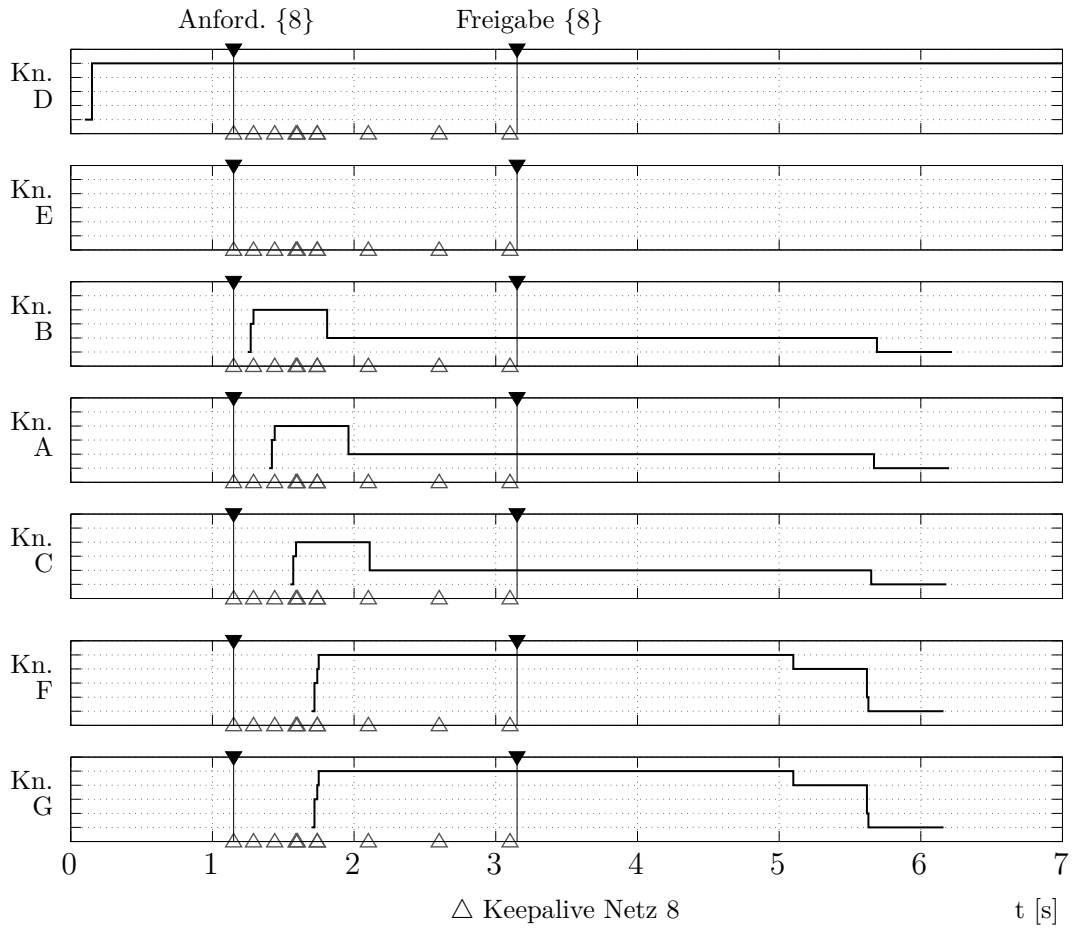
Anmerkung: Die Knotenzustände auf der Ordinate sind OFFLINE (1), BRIDGE (2), REMOTE WAKE (3), NO NETWORK (4), NETWORK (5)

D.5 Zusammenwachsen zweier Teilnetze



Anmerkung: Die Knotenzustände auf der Ordinate sind OFFLINE (1), BRIDGE (2), REMOTE WAKE (3), NO NETWORK (4), NETWORK (5)

D.6 Anfordern eines entfernten Teilnetzes



Anmerkung: Die Knotenzustände auf der Ordinate sind OFFLINE (1), BRIDGE (2), REMOTE WAKE (3), NO NETWORK (4), NETWORK (5)

Literaturverzeichnis

[1] A. Vollmer: *Deutsche OEMs setzen Standards*. http://www.all-electronics.de/deutsche-oems-setzen-standards, Abruf: 07.09.2016

[2] Advanced Micro Devices, Inc.: *Magic Packet Technology*. http://support.amd.com/TechDocs/20213.pdf. Version: 1995, Abruf: 06.07.2016. – Whitepaper

[3] American National Standards Institute: *ANSI X3.263-1995: Fibre Distributed Data Interface (FDDI) - Token Ring Twisted Pair Physical Layer Medium Dependent (TP-PMD)*. 1995

[4] AUTOSAR: *Website*. http://www.autosar.org, Abruf: 06.07.2016

[5] AUTOSAR: *Specification of CAN Network Management*. Release 3.0, 2010

[6] AUTOSAR: *Specification of CAN State Manager*. Release 3.0, 2010

[7] AUTOSAR: *Specification of Communication Manager*. Release 3.0, 2010

[8] AUTOSAR: *Specification of ECU State Manager*. Release 3.0, 2010

[9] AUTOSAR: *Specification of FlexRay Network Management*. Release 3.0, 2010

[10] AUTOSAR: *Specification of NetworkManagement Interface*. Release 3.0, 2010

[11] AUTOSAR: *Layered Software Architecture*. http://www.autosar.org/fileadmin/files/releases/4-2/software-architecture/general/auxiliary/AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf. Version: Release 4.2.2, 2011, Abruf: 06.07.2016

[12] AUTOSAR: *Specification of CAN Network Management*. Release 4.2.2, 2015

[13] AUTOSAR: *Specification of Communication Manager*. Release 4.2.2, 2015

[14] AUTOSAR: *Specification of ECU State Manager*. Release 4.2.2, 2015

[15] AUTOSAR: *Specification of NetworkManagement Interface*. Release 4.2.2, 2015

[16] AUTOSAR: *Specification of UDP Network Management*. Release 4.2.2, 2015

[17] Braess, H. (Hrsg.) ; Seiffert, U. (Hrsg.): *Vieweg-Handbuch Kraftfahrzeugtechnik: Mit 50 Tabellen*. 7. aktualisierte Aufl. Wiesbaden : Springer Vieweg, 2013 (ATZ/MTZ-Fachbuch). – ISBN 3658016914

[18] Broadcom: *53115M Multiport Gigabit Ethernet Switches*. 2012. – Datenblatt

[19] Broadcom Corporation: *BroadR-Reach Physical Layer Transceiver Specification For Automotive Applications*. `http://www.ieee802.org/3/1TPCESG/public/BroadR_Reach_Automotive_Spec_V3.0.pdf`. Version: 2014, Abruf: 06-07-2016

[20] CAN in Automation e.V.: *Website*. `http://www.can-cia.org`, Abruf: 06.07.2016

[21] Case, J. ; McCloghrie, K. ; Rose, M. ; Waldbusser, S.: *Introduction to version 2 of the Internet-standard Network Management Framework*. RFC 1441 (Historic). `http://www.ietf.org/rfc/rfc1441.txt`. Version: April 1993 (Request for Comments)

[22] Case, J. ; Mundy, R. ; Partain, D. ; Stewart, B.: *Introduction and Applicability Statements for Internet-Standard Management Framework*. RFC 3410 (Informational). `http://www.ietf.org/rfc/rfc3410.txt`. Version: Dezember 2002 (Request for Comments)

[23] Case, J.D. ; Fedor, M. ; Schoffstall, M.L. ; Davin, J.: *Simple Network Management Protocol (SNMP)*. RFC 1157 (Historic). `http://www.ietf.org/rfc/rfc1157.txt`. Version: Mai 1990 (Request for Comments)

[24] Christensen, K. ; Reviriego, P. ; Nordman, B. ; Bennett, M. ; Mostowfi, M. ; Maestro, J.: IEEE 802.3az: the road to energy efficient ethernet. In: *IEEE Communications Magazine* 48 (2010), Nr. 11

[25] eNOVA Strategiekreis Elektromobilität: *SEIS - Sicherheit in Eingebetteten, IP-Basierten Systemen*. `http://www.strategiekreis-elektr\omobilitaet.de/public/projekte/seis`. Version: 2012, Abruf: 06.07.2016

[26] Esch, S. ; Meyer, J. ; Linn, G.: Partial Networking: Deactivation of Inactive Ecus. In: *ATZelektronik worldwide* 7 (2012), Nr. 1

[27] Esch, S. ; Meyer, J. ; Linn, G.: Teilnetzbetrieb: Abschaltung Inaktiver Steuergeräte. In: *ATZelektronik* 7 (2012), Nr. 1

[28] Europäische Kommission: *Mittelung KOM(2011) 112 endg.: Fahrplan für den Übergang zu einer wettbewerbsfähigen CO2-armen Wirtschaft bis 2050.* 2011. – Mitteilung der Europäischen Kommission

[29] Europäische Kommission: *Weissbuch KOM(2011) 144 endg.: Fahrplan zu einem einheitlichen europäischen Verkehrsraum - Hin zu einem wettbewerbsorientierten und ressourcenschonenden Verkehrssystem.* 2011

[30] Europäisches Parlament: *Verordnung (EG) Nr. 443/2009 des Europäischen Parlaments und des Rates vom 23. April 2009 zur Festsetzung von Emissionsnormen für neue Personenkraftwagen im Rahmen des Gesamtkonzepts der Gemeinschaft zur Verringerung der CO2-Emissionen von Personenkraftwagen und leichten Nutzfahrzeugen.* 2009. – Verordnung des Europäischen Parlaments

[31] FlexRay Consortium: *FlexRay Communications System, Electrical Physical Layer Specification.* 2006

[32] Fock, F.: *AGENT++ SNMP Software.* 2009

[33] Freescale Semiconductor, Inc.: *MPC5668x Microcontroller Reference Manual.* 2011

[34] Grzemba, A.: *MOST: Das Multimedia-Bussystem für den Einsatz im Automobil.* 1. Aufl. Franzis, 2007 (Elektronik & Elektrotechnik Bibliothek). – ISBN 9783772341496

[35] Gupta, M. ; Singh, S.: Greening of the Internet. In: *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications.* New York : ACM, 2003 (SIGCOMM '03). – ISBN 1581137354

[36] IEEE 802.1: *Time Sensitive Networking Task Group.* 2016

[37] IEEE 802.3 Ethernet Working Group: *Call for Interest, 1 Twisted Pair 100 Mbit/s Ethernet (1TPCE).* `http://www.ieee802.org/3/cfi/0314_2/CFI_02_0314.pdf`. – IEEE 802 Plenary Meeting Beijing, China, März 2014

[38] IEEE 802.3 Ethernet Working Group: *Call for Interest, Reduced Twisted Pair Gigabit Ethernet PHY.* `http://www.ieee802.org/3/RTPGE/public/mar12/CFI_01_0312.pdf`. – IEEE 802 Plenary Meeting Orlando, USA, März 2012

[39] IEEE 802.3bp: *1000BASE-T1 PHY Task Force.* `http://www.ieee802.org/3/bp/`, Abruf: 08.07.2016

[40] IEEE Standards Association: *IEEE Std 802.3-2012: IEEE Standard for Ethernet*. 2012

[41] IEEE Standards Association: *IEEE Std 802.1Q-2014: IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks*. 2014

[42] IEEE Standards Association: *IEEE 802.1Qbv-2015 - IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic*. 2015

[43] IEEE Standards Association: *IEEE Std 802.3bw-2015: IEEE Standard for Ethernet Amendment: Physical Layer Specifications and Management Parameters for 100 Mb/s Operation over a Single Balanced Twisted Pair Cable (100BASE-T1)*. 2015

[44] Intel: *Intel 82574 GbE Controller Family*. `http://www.intel.de/content/dam/doc/datasheet/82574l-gbe-controller-datasheet.pdf`. Version: Rev. 3.4, 2014, Abruf: 06.07.2016. – Datenblatt

[45] International Standards Organization: *ISO 17356:2005, Road vehicles - Open interface for embedded applications*. 2005

[46] International Standards Organization: *ISO 17356-5:2006, Road vehicles - Open interface for embedded applications - Part 5: OSEK/VDX Network Management*. 2006

[47] International Standards Organization: *ISO 11898-6:2013, Road vehicles - Controller Area Network (CAN) - Part 6: High-speed medium access unit with selective wake-up functionality*. 2013

[48] International Standards Organization: *ISO 11898:2015, Road vehicles - Controller Area Network (CAN)*. 2015

[49] International Standards Organization: *ISO 17987, Road vehicles - Local Interconnect Network (LIN)*. 2016

[50] Kunze, S. ; Pöschl, R. ; Grzemba, A.: Comparison of Energy Optimization Methods for Automotive Ethernet Using Idealized Analytical Models. In: Schulze, T. (Hrsg.) ; Müller, B. (Hrsg.) ; Meyer, G. (Hrsg.): *Advanced Microsystems for Automotive Applications 2015: Smart Systems for Green and Automated Driving*. Cham : Springer, 2016. – ISBN 9783319208558

[51] LIN Konsortium: *Website*. `http://www.lin-subbus.org`, Abruf: 06.07.2016

[52] Lohse, D. ; Schnabel, W.: *Grundlagen der Straßenverkehrstechnikund der Verkehrsplanung*. 1. Aufl. Berlin : Beuth-Verlag, 2011. – ISBN 9783410203995

[53] Massa, A. J.: *Embedded software development with eCos*. Upper Saddle River and NJ : Prentice Hall, 2003 (Bruce Perens' open source series). – ISBN 9780130354730

[54] Matheus, K. ; Königseder, T.: *Automotive Ethernet*. 1. Aufl. Cambridge, U.K. : Cambridge Univ. Pr., 2015. – ISBN 1107057280

[55] McCloghrie, K. ; Kastenholz, F.: *The Interfaces Group MIB*. RFC 2863 (Draft Standard). `http://www.ietf.org/rfc/rfc2863.txt`. Version: Juni 2000 (Request for Comments)

[56] McCloghrie, K. ; Perkins, D. ; Schoenwaelder, J.: *Structure of Management Information Version 2 (SMIv2)*. RFC 2578 (INTERNET STANDARD). `http://www.ietf.org/rfc/rfc2578.txt`. Version: April 1999 (Request for Comments)

[57] McCloghrie, K. ; Rose, M.: *Management Information Base for Network Management of TCP/IP-based internets: MIB-II*. RFC 1213 (INTERNET STANDARD). `http://www.ietf.org/rfc/rfc1213.txt`. Version: März 1991 (Request for Comments). – Updated by RFCs 2011, 2012, 2013

[58] McCloghrie, K. ; Rose, M.T.: *Management Information Base for network management of TCP/IP-based internets*. RFC 1156 (Historic). `http://www.ietf.org/rfc/rfc1156.txt`. Version: Mai 1990 (Request for Comments)

[59] Merker, G. ; Schwarz, P.: *Grundlagen Verbrennungsmotoren: Funktionsweise, Simulation, Messtechnik*. 6. ergänzte Aufl. Wiesbaden : Vieweg+Teubner, 2012 (ATZ/MTZ-Fachbuch). – ISBN 3834819883

[60] Metz, B. (Hrsg.) ; Davidson, O. (Hrsg.) ; Bosch, P. (Hrsg.) ; Dave, R. (Hrsg.) ; Meyer, L. (Hrsg.): *Climate change 2007 - Mitigation of climate change: Working group III contribution to the fourth assessment report of the IPCC*. Cambridge U.K. : Cambridge Univ. Pr., 2007. – ISBN 9780521880114

[61] Mick, C. ; DiMinico, C. ; Raghavan, S. ; Rao, S. ; Hatamian, M.: *IEEE Std 802.3ab 1000BASE-T Tutorial Presentation*. Version: 1998. `http://grouper.ieee.org/groups/802/3/tutorial/march98/mick_170398.pdf`, Abruf: 07.06.2016

[62] MOST Cooperation: *Website*. http://www.mostcooperation.com, Abruf: 06.07.2016

[63] NXP Semiconductors: *TJA1145: High-speed CAN transceiver for partial networking*. http://www.nxp.com/documents/data_sheet/TJA1145.pdf. Version: Rev. 02, 2014, Abruf: 06.07.2016. – Datenblatt

[64] NXP Semiconductors: *TJA1041: High speed CAN transceiver*. http://www.nxp.com/documents/data_sheet/TJA1041.pdf. Version: Rev. 06, 2007, Abruf: 06.07.2016. – Datenblatt

[65] Olivier, J. ; Janssens-Maenhout, G. ; Muntean, M. ; Peters, J.: *JRC reference report*. Bd. JRC83593: *Trends in Global CO-2 Emissions: 2013 Report*. Den Haag : Amt für Veröffentlichungen der Europäischen Union, 2013. – ISBN 9789491506512

[66] OPEN Alliance SIG: *One Pair Ethernet Alliance Special Interest Group*. 2016

[67] Ortega, L. D. ; Kraft, K. H. ; Claus, L.: Untersuchung von passiven FlexRay-Sternkopplern. In: *Elektronik automotive* (2007), Nr. 7

[68] Presuhn, R.: *Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)*. RFC 3418 (INTERNET STANDARD). http://www.ietf.org/rfc/rfc3418.txt. Version: Dezember 2002 (Request for Comments)

[69] Rausch, M.: *FlexRay: Grundlagen, Funktionsweise, Anwendung*. Hanser Fachbuchverlag, 2007. – ISBN 9783446412491

[70] Reif, K.: *Automobilelektronik: Eine Einführung für Ingenieure ; 36 Tabellen*. 3. überarbeitete Aufl. Wiesbaden : Vieweg+Teubner, 2009 (ATZ/MTZ-Fachbuch). – ISBN 9783834804464

[71] Reviriego, P. ; Hernandez, J. A. ; Larrabeiti, D. ; Maestro, J. A.: Performance evaluation of energy efficient ethernet. In: *IEEE Communications Letters* 13 (2009), September, Nr. 9

[72] Schmutzler, C.: *Hardwaregestützte Energieoptimierung von Elektrik/Elektronik-Architekturen durch adaptive Abschaltung von verteilten, eingebetteten Systemen*, Karlsruher Institut für Technologie, Diss., 2012

[73] Schmutzler, C. ; Krüger, A. ; Schuster, F. ; Simons, M.: Energy efficiency in automotive networks: Assessment and concepts. In: *International Conference on High Performance Computing and Simulation (HPCS), 2010*, 2010

[74] Seyler, J. R. ; Streichert, T. ; Warkentin, J. ; Spägele, M. ; Glass, M. ; Teich, J.: A self-propagating wakeup mechanism for point-to-point networks with partial network support. In: *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2014

[75] Siebenpfeiffer, W. (Hrsg.): *Vernetztes Automobil*. 1. Aufl. Wiesbaden : Springer Vieweg, 2014 (ATZ/MTZ-Fachbuch). – ISBN 9783658040185

[76] SMSC: *Datenblatt Netzwerk-Controller OS81050*. `http://www.smsc.com`. Version: Juli 2012, Abruf: 06.07.2016

[77] Stan, C.: *Thermodynamik des Kraftfahrzeugs*. 2. Aufl. Berlin/Heidelberg : Springer, 2012. – ISBN 9783642276309

[78] Suermann, T. ; Müller, S.: Power Saving in Automotive Ethernet. In: Fischer-Wolfarth, J. (Hrsg.) ; Meyer, G. (Hrsg.): *Advanced Microsystems for Automotive Applications 2014: Smart Systems for Safe, Clean and Automated Vehicles*. Cham : Springer, 2014. – ISBN 9783319080871

[79] Varga, A.: *OMNeT++ Discrete Event Simulator*. 2015. – Version 4.6

[80] Varga, A. ; Hornig, R.: An Overview of the OMNeT++ Simulation Environment. In: *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*. Brussels, Belgium : ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008. – ISBN 9789639799202

[81] VDA Verband der Automobilindustrie e.V.: *Handeln für den Klimaschutz: CO2 Reduktion in der Automobilindustrie*. 2. überarbeitete Aufl. August 2009

[82] Weber, R. ; Watroba, E. ; Schmitz, C.: CO2 reduction through Can partial networking. In: *ATZelektronik worldwide* 6 (2011), Nr. 2

[83] Weiss, M.: *JRC reference report*. Bd. JRC62639: *Analyzing on-road emissions of light-duty vehicles with portable emission measurement systems (PEMS)*. Luxembourg : Amt für Veröffentlichungen der Europäischen Union, 2011. – ISBN 9279190725

[84] Wille, M.: Quickly Supporting Market Needs shown by Partial Networking. In: *ATZextra worldwide* (2013), Nr. 9

[85] Zimmermann, W. ; Schmidgall, R.: *Bussysteme in der Fahrzeugtechnik*. 2. aktualisierte und erweiterte Aufl. Wiesbaden : Vieweg+Teubner, 2007 (ATZ/MTZ-Fachbuch). – ISBN 3834819883

[86] Zinner, H.: *Vernetzung heterogener Feldbusse auf Basis des Standards Ethernet Audio Video Bridging*, Technische Universität Ilmenau, Diss., 2015

Eigene Veröffentlichungen

[87] Balbierer, N.: *Ein Energiemanagement-Konzept für den Einsatz von IP-Ethernet im Bereich Automotive.* Deutschland, Hochschule Regensburg, Diplomarbeit, September 2009

[88] Balbierer, N.: Energieeffizienz bei auf IP und Ethernet basierenden Fahrzeugnetzen. In: *11. Ilmenauer TK-Manager Workshop.* Ilmenau, Deutschland : Univ.-Verl. Ilmenau, 2011. – ISBN 9783939473978

[89] Balbierer, N.: *Energieeffiziente Fahrzeugnetze.* September 2012. – 57. Internationales Wissenschaftliches Kolloquium, Ilmenau, Deutschland

[90] Balbierer, N.: *Ethernet Partial Networking: Concept and Realization.* November 2014. – Hanser Automotive Networks, Stuttgart, Deutschland

[91] Balbierer, N. ; Nöbauer, J. ; Kern, A. ; Dremel, B. ; Camek, A. ; Heinrich, P. ; Weiler, H.: *Netzwerkmanagement bei IP-basierten Netzwerken im Automobil.* September 2011. – SEIS Statusseminar, München, Deutschland, `http://www.strategiekreis-elektromobilitaet.de/public/projekte/seis/das-sichere-ip-basierte-fahrzeugbordnetz/pdfs/TP2_Vortrag3.pdf`, [Online; Stand 04. Juli 2016]

[92] Balbierer, N. ; Waas, T.: *Teilnetzbetrieb bei IP-basierten Fahrzeugnetzwerken.* Februar 2012. – 4. Elektronik Automotive Kongress, München, Deutschland

[93] Balbierer, N. ; Waas, T. ; Meyer, J. ; Jakob, M. ; Seitz, J.: Multinet - A wake-up protocol for multicast network groups. In: *Proceedings of the 11th IEEE Workshop on Intelligent Solutions in Embedded Systems (WISES)*, 2013

[94] Balbierer, N. ; Waas, T. ; Nöbauer, J. ; Seitz, J.: Energy consumption of Ethernet compared to automotive bus networks. In: *Proceedings of the Ninth Workshop on Intelligent Solutions in Embedded Systems (WISES)*, 2011

[95] Balbierer, N. ; Zinner, H.: *Ethernet als energieeffizientes Fahrzeugnetzwerk.* Juli 2012. – SEIS Abschlussveranstaltung, Wolfsburg, Deutschland

[96] Balbierer, N. ; Zinner, H. ; Nöbauer, J. ; Gallner, T.: *Ethernet in Automotive applications.* März 2010. – Fully Networked Car @ Geneva Motorshow, Genf, Schweiz

[97] Nöbauer, J. ; Balbierer, N. ; Zinner, H. ; Meier, H.: *Verbundvorhaben: SEIS (Sicherheit in eingebetteten IP-basierten Systemen): Teilvorhaben: Steuergeräte und elektromechanische Fahrzeugkomponenten für IP-fähige Bordnetze; Abschlussbericht.* Juni 2012. – Technische Informationsbibliothek Hannover

[98] Strobl, M. ; Waas, T. ; Moehne, S. ; Kucera, M. ; Rath, A. ; Balbierer, N. ; Schingale, A.: Using Ethernet over powerline communication in automotive networks. In: *Proceedings of the 10th IEEE Workshop on Intelligent Solutions in Embedded Systems (WISES)*, 2012

Patente und Patentanmeldungen

[99] Balbierer, N. ; Nöbauer, J.: *Verfahren zur Aktivierung einer Netzwerk-Komponente eines Fahrzeug-Netzwerksystems.* – Offenlegungsschrift DE102010008818, Offenlegungstag: 25. August 2011, Deutsches Patentamt

[100] Balbierer, N. ; Nöbauer, J.: *Verfahren zur Aktivierung von deaktivierten Steuergeräten eines Fahrzeugs und Fahrzeugnetz sowie Knoten des Fahrzeugnetzes.* – Offenlegungsschrift DE102012207858, Offenlegungstag: 14. November 2013, Deutsches Patentamt

[101] Balbierer, N. ; Nöbauer, J. ; Zinner, H.: *Verfahren zum Übertragen von Daten mit einem Ethernet-AVB-Transportprotokoll zwischen Knoten eines Kraftfahrzeugs sowie zur Durchführung des Verfahrens eingerichtetes Steuergerät.* – Offenlegungsschrift DE102012207883, Offenlegungstag: 14. November 2013, Deutsches Patentamt

[102] Balbierer, N. ; Nöbauer, J. ; Zinner, H.: *Verfahren zum Übertragen von Daten mit einem Ethernet-Transportprotokoll zwischen Knoten eines Kraftfahrzeugs sowie zur Durchführung des Verfahrens eingerichtetes Steuergerät.* – Offenlegungsschrift DE102012207900, Offenlegungstag: 14. November 2013, Deutsches Patentamt

[103] Balbierer, N. ; Röder, J.: *Weckschaltung für eine an einem Zweidrahtbus für ein differentielles Signal betreibbare Buskomponente und Buskomponente mit einer solchen Weckschaltung.* – Offenlegungsschrift DE102013208004, Offenlegungstag: 10. Juli 2014, Deutsches Patentamt

Betreute studentische Arbeiten

[104] Bartz, B.: *Energy efficient management concept based on IP/Ethernet in automotive environment*. Deutschland, Hochschule Regensburg, Diplomarbeit, März 2010

[105] Lammering, D.: *Integration eines eCos-Betriebssystems und Implementierung eines CAN-Ethernet-Tunneling mit Restbussimulation auf einem eingebetteten System im Automobil*. Deutschland, Hochschule Regensburg, Masterarbeit, Oktober 2013

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ADAS ..... Advanced Driver Assistance Systems
ASG ..... Anhängersteuergerät
AUTOSAR ..... AUTomotive Open System ARchitecture
BGEM ..... Border GEM
BSW ..... AUTOSAR Basic Software
CAN ..... Controller Area Network
CFI ..... Call for Interest
ComM ..... AUTOSAR Communication Manager
CSMA/CR ..... Carrier Sense Multiple Access/Collision Resolution
E/E ..... Elektrik/Elektronik
eCos ..... Embedded Configurable Operating System
ECU ..... Electronic Control Unit
EcuM ..... AUTOSAR ECU State Manager
EDM ..... Energy Detect Modul
EEE ..... Energy Efficient Ethernet
EU ..... Europäische Union
FOT ..... Fiber Optical Transceiver
GEM ..... Gigabit Ethernet Modul
GMII ..... Gigabit Media Independent Interface
HF ..... Hochfrequenz
HSG ..... Heckdeckelsteuergerät
IEEE ..... Institute of Electrical and Electronics Engineers
INIC ..... Intelligent Network Interface Controller
IP ..... Internet Protokoll
IPCC ..... Intergovernmental Panel on Climate Change

ISO ..... International Standards Organization
KERS ..... Kinetic Energy Recovery System
LAN ..... Local Area Network
LIN ..... Local Interconnect Network
LPI ..... Low Power Idle
LVDS ..... Lov Voltage Differential Signaling
MCLI ..... Multinet Client
MDIO ..... Management Data Input/Output
MII ..... Media Independent Interface
MOSFET ..... Metal Oxide Semiconductor Field Effect Transistor
MOST ..... Media Oriented Systems Transport
NEFZ ..... Neuer Europäischer Fahrzyklus
NM ..... Netzwerkmanagement
NRZ ..... Non Return to Zero
OSEK ..... Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug
OSI ..... Open Systems Interconnect
PCIe ..... Peripheral Component Interconnect Express
PDU ..... Protocol Data Unit
PHY ..... Physical Layer
PKW ..... Personenkraftwagen
PMA ..... Physical Medium Attachment Sublayer
PMD ..... Physical Medium Dependent Sublayer
PNC ..... Partial Network Cluster
POSIX ..... Portable Operating System Interface
RTE ..... AUTOSAR Runtime Environment
RTPGE ..... Reduced Twisted Pair Gigabit Ethernet
RvMII ..... Reverse Media Independent Interface
SHL ..... Sitzssteuergerät hinten links
SHR ..... Sitzssteuergerät hinten rechts
SIM ..... Botschaftssimulator
SPICE ..... Simulation Program with Integrated Circuit Emphasis

SVL ..... Sitzssteuergerät vorne links
SVR ..... Sitzssteuergerät vorne rechts
TDM ..... Time Domain Multiplexing
TL ..... Türsteuergerät links
TR ..... Türsteuergerät rechts
TTL ..... Time to live
UDP ..... User Datagram Protocol
VHL ..... Ventilsteuergerät hinten links
VHR ..... Ventilsteuergerät hinten rechts
VVL ..... Ventilsteuergerät vorne links
VVR ..... Ventilsteuergerät vorne rechts
WOL ..... Wake on LAN