Seminar Sommersemester 2009 Automotive Konzepte und Techniken
Prof. Dr. Dieter Zöbel, Universität Koblenz-Landau, FB Informatik

CAN Controller Area Network
Analyse, Bandbreite und Optimierung

Christel-Joy Cameran

Eingereicht: 1.7.2009 / Fertiggestellt: 24.7.2009

Zusammenfassung Der Controller Area Network (CAN) gehört zu den etabliertesten Netzwerken in der Automobil-Branche. Um weiterhin die auf dem Markt immer steigenden Anforderungen an Brandbreite erfüllen zu können, ist eine Lösung die Optimierung durch Offsets. Dafür werden wir die generelle Funktionsweise des CAN-Busses erläutern, dazu gehören das CSMA/CA Verfahren wie auch die Arbitrierung. Außerdem wird ein Überblick über die verschiedenen Fehlererkennungs- und Fehlerbehandlungs-Mechanismen gezeigt. Um eine optimale Setzung der Offsets zu gewährleisten werden wir verschiedene Scheduling Modelle analysieren wie auch die maximale Antwortzeit oder auch Worst-Case Response Time (WCRT) genannt. Letzt- endlich werden wir die WCRT minimieren und dadurch an Performanz dazu gewinnen, indem wir die Auslastung des Busses zeitlich durch Offsets streuen. Damit soll eine kostengünstige Alternative zu anderen Netzwerken wie FlexRay geboten werden.

Schlüsselwörter CSMA/CA · Frame-Arbitrierung · Fehlererkennung und Fehlerbehandlung · Transmission Time · Scheduling · WCRT · Offset

1 Einführung

1.1 Motivation

Die Verfügbarkeit kostengünstiger Komponenten der Mikroelektronik führte in der Kraftfahrzeugtechnik zu Beginn der achtziger Jahre zur Einführung von weitgehend autonomen elektronischen Steuergeräten für unterschiedliche Funktionsbereiche wie z.B. Zündungs-, Getriebesteuerungs- oder Antiblockiersysteme. Herbei zeigte sich bald, dass weitere Funktionsverbesserungen und damit eine signifikante Verbesserung des Fahrverhaltens erst dann möglich sein würde, wenn eine Zusammenarbeit der auf den verschiedenen Steuergeräten verteilten Prozesse durch Datenaustausch zwischen den Geräten möglich ist. Wegen der besonders hohen Anforderungen an die Sicherheit der Datenübertragung in einer von elektromagnetischen Störungen gekennzeichneten

Christel-Joy Cameran
University Koblenz-Landau
E-Mail: cameran@uni-koblenz.de

Umgebung war es notwendig, ein besonders für diesen Einsatz geeignetes Kommunikationskonzept zu entwickeln, da die bereits vorhandenen, für andere Einsatzbereiche entwickelten Lösungen, diese nicht erfüllen konnten.

1.2 Geschichte

Dies war die Ausgangslage für die, von der Firma Bosch begonnene Entwicklung des Controller Area Network (CAN) Protokoll im Jahre 1983. Mit Intel zusammen wurde der asynchrone serielle CAN-Bus 1985 zur Vernetzung von Steuergeräten in Automobilen vorgestellt. Hauptsächlich sollte eine Reduktion der Kabel herbeigeführt werden, um Gewicht und zeitgleich Kosten zu sparen. Aber auch die Situation an neuralgischen Verkabelungsstellen, wie etwa die der Kabeldurchführung vom Innenraum zu den Vordertüren konnten so entschärft werden.

1987 wurde die Schnittstelle fertig gestellt. Ein Jahr später stand der Chip von Intel zur Verfügung und Anfang der 90er Jahre schaffte es den Durchbruch im Automobilbau. Der CAN-Bus wurde 1991 erstmals in der Mercedes S-Klasse serienmäßig verbaut und ist seit 2001 auch in Kleinwagen zu finden.

1.3 Einsatzbereich

Durch die hohe Verbreitung von CAN sind die Komponenten sehr preisgünstig und in hohen Stückzahlen erhältlich. Der Feldbus hat unterschiedliche Einsatzbereiche, da seine Geschwindigkeit recht flexibel von 10kbit/s bis 1Mbit/s reicht. Meistens werden zwei getrennte CAN-Busse verwendet. Man unterschiedet sie entsprechend den erforderlichen Datenraten in Highspeed-Bus (größer 250 kbit/s) für die Motor und Sicherheitssteuerung und in Lowspeed-Bus (kleiner 125 kbit/s) für die Komfort Elektronik. Die Buslänge bei CAN steht unmittelbar im Zusammenhang mit der möglichen maximalen Arbeitsgeschwindigkeit. Der Standardwert hierfür liegt bei 125 kbit/s und erlaubt Buslängen bis 500m. Beide Systeme sind voneinander getrennt. Ein Datenaustausch zwischen beiden Systemen ist über einen Gateway möglich. Im Zuge der immer weiter fortschreitender Audio- und Videotechnik in Automobilen wird mittlerweile neben dem CAN-Bus (und weiteren) noch ein Multimedia-Bus betrieben, namens MOST.

Mittlerweile hat sich die Nutzung des CAN-Busses nicht nur in Personen- und Nutzfahrzeugen sondern auch im gesamten Bereich mobiler Systeme durchgesetzt. Er findet einen vielfältigen Einsatz auch in der Industrieautomation wie z.B. in Aufzügen, landwirtschaftlichen Maschinen, Kränen, Baumaschinen, Müllfahrzeugen, militärischen Fahrzeugen oder im öffentlichen Nahverkehr. [*O. Alt, K. Etschberger*]

CAN ist das führende serielle Bus System für eingebettete Controller und ein etabliertes Netzwerk, dass 2003 international standardisiert wurde (ISO 11898-1). [*W. Zimmermann, R. Schmidgall, S. 107*]

2 Bandbreite und Frames

CAN basiert auf einer linienförmigen oder sternförmigen Topologie und auf der Multi-Master-Architektur. Die Kommunikation erfolgt ereignisgesteuert. Beim Standard-CAN können maximal $2^{11} = 2048$ Identifier (11 Bits = Länge des Identifiers) verschiedene Nachrichten-Identifier definiert werden. Alle Teilnehmer am Bus hören das gleiche Bit und, da die Reaktion aller Steuergeräte innerhalb einer Bitzeit erfolgt, muss die Buslänge um so kleiner sein, je höher die Bitrate ist. Die maximale Buslänge ist 40 m bei 1 MBit/s (1000m bei 500 kbit/s).

Es werden 4 Frame Formate unterschieden: Data, Remote, Error und Overload Frame. Über das Data Frame (Datentelegramm) erfolgt die Datenübertragung von einem Sender zu einem oder mehreren Empfängern auf Initiative der Datenquelle, also dem Sender. Mit dem Remote Frame (Datenanforderungstelegramm) können Busteilnehmer, genauer gesagt die Empfänger, das Senden eines bestimmten Datentelegramms mit demselben Identifier durch eine Datenquelle anfordern. Durch das Error Frame (Fehlertelegramm) erfolgt die Signalisierung eines von einem Busteilnehmer, also Sender oder Empfänger, erkannten Fehlers. Und das Overload Frame (Überlasttelegramm) kann eine Verzögerung zwischen einem vorangegangenen und einem nachfolgenden Data oder einem Remote Frame realisiert werden. Sie können mit einem Abstand (Interframe Space) von mindestens 3 Bits aufeinander folgen.

2.1 Data Frame

Ein Data Frame setzt sich aus sieben Feldern zusammen. Das Startfeld (1Bit) markiert den Beginn eines Frames durch ein dominantes Bit. Das Arbitrierungsfeld (12 Bits) fügt sich aus dem Identifier (11Bits) und dem RTR-Bit zusammen. Das RTR-Bit dient zur Unterscheidung zwischen Data und Remote Frame. Beim Data Frame ist ein dominantes Bit. Das Kontroll- oder Steuerfeld (6 Bits) besteht aus 4 Bits, die die Länge des Datenfelds angeben und 2 Bits die für Erweiterungen reserviert sind. Das Datenfeld (0-64 Bits) enthält die eigentliche Nutzinformation einer CAN-Nachricht und kann 0 bis 8 Byte umfassen. Das Datensicherungsfeld (CRC-Feld, 16 Bits) besteht aus einer 15 Bits Prüfsequenz sowie einem rezessiv übertragenen Begrenzungsbit. Mit der, in der Prüfsequenz enthaltenen redundanten Information kann ein Empfänger nachprüfen, ob die empfangene Nachricht durch Störeinwirkungen verfälscht wurde. Das Bestätigungsfeld (Acknowledge Feld, 2 Bits) setzt sich aus dem Acknowledge Slot und einem Begrenzungsbit zusammen und dient als Empfangsbestätigung vom Empfänger. Das Endefeld (7 Bits) schließt das Frame mit 7 rezessiven Bits ab.

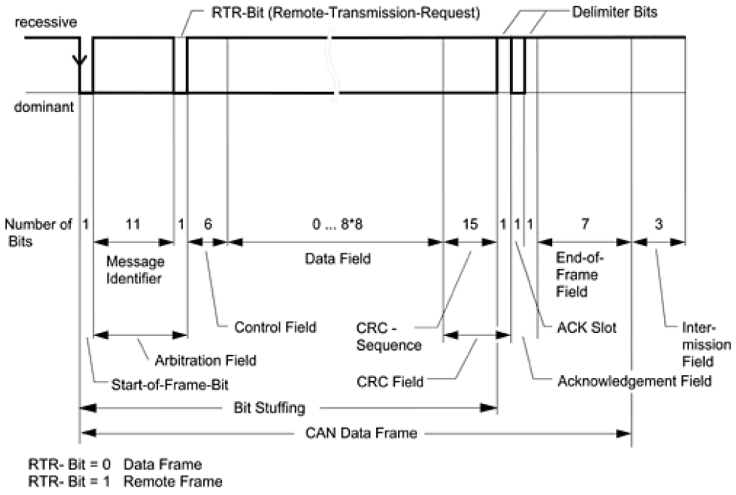

Abb. 1 *Format einer Standard CAN-Nachricht (Data Frame). Das Format einer Anforderungsnachricht (Remote Frame) unterscheidet sich lediglich durch den Wert des RTR-Bits sowie einem leeren Datenfeld.*

3 Frame-Arbitrierung

Ein wichtiger Punkt bei CAN ist die stattfindende Frame-Arbitrierung. Der Bus arbeitet nach dem CSMA/CA (Carrier Sense Multiple Access / Collision Avoidance) Verfahren und ermöglicht so einen kollisionsfreien Buszugriff, wie auch eine durch Prioritäten gesteuerte Gewichtung der Daten. Hierbei ist jeder Teilnehmer bezüglich des Buszugriffs gleichberechtigt (Multi-Master-System) und kann auf den Bus zugreifen, sobald dieser frei ist (zufälliger Buszugriff). Das Verfahren ermöglicht eine ereignisgesteuerte Datenübertragung, die nur dann initiiert wird, wenn dies vom Sender aus als erforderlich angesehen wird.

Wie bei jedem anderen ereignisgesteuerten System ist auch der CAN-Bus nicht deterministisch. Der bei kontrollierten Verfahren erforderliche Busverkehr für die Verwaltung des Buszugriffs entfällt dadurch. Außerdem resultieren hieraus erheblich niedrigere mittlere Busbelastungen sowie eine sehr kurze Latenzzeit gegenüber zyklischen Verfahren. Da aber nach Freiwerden des Busses auch mehrere Teilnehmer gleichzeitig den Bus benutzen könnten, würde dies zu einer Kollision führen. Hier greift die bitweise Arbitrierung über den Identifier ein. Die Kommunikation über Nachrichtenidentifikation unterscheidet priorisierte Nachrichten und ermöglicht so am Ende der Arbitrierungsphase, dass nur ein Busteilnehmer den Bus belegt und so seine Nachricht verlustfrei übertragen kann. Legt man darüber hinaus fest, dass eine hoch priorisierte Nachricht nicht ständig den Bus anfordern kann, so ist es trotz des zufälligen Buszugriffs möglich, Aussagen über die Latenzzeit einer Nachricht zu machen und somit die allgemeine Bedingung für Echtzeitfähigkeit zu erfüllen.

Es gibt keine Empfängeradressen im Bus, so dass jeder Teilnehmer alle Nachrichten hört (Broad-/Multicast). Diese Fähigkeit ist unter anderem sehr nützlich für die Synchronisation von Anwendungs- prozessen. Ein Knoten im CAN-Netzwerk kann entweder im Sende- oder im Empfangsmodus sein. Während ein Knoten sendet, lesen alle anderen Knoten vom Bus. Die Empfänger entscheiden anhand des Identifiers einer

Nachricht, ob diese für sie relevant ist oder nicht. Nach dem Ende der Nachricht kann der Knoten mit der am höchsten priorisierten Nachricht auf den Bus senden.

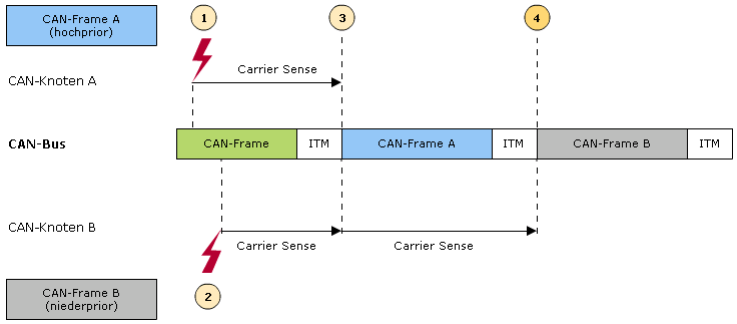

Abb. 2 *1. Knoten A, Nachricht mit höherer Priorität, will auf den Bus zugreifen*
2. Knoten B, Nachricht mit niedriger Priorität, will auf den Bus zugreifen
3. Beide Knoten senden, aber Knoten B muss wieder warten
4. Nun kann auch Knoten B senden

3.1 Buspegel

Grundlage der bitweisen Arbitrierung ist die Unterscheidung von zwei physikalischen Buspegeln, nämlich einem dominanten (überstimmenden) und einem rezessiven (nachgebeden) Pegel. Wollen nun beispielsweise mehrere Steuergeräte einen Datenframe senden, so geben sie jeweils ihrem CAN-Controller den Auftrag dazu. Jeder Controller schreibt nun zunächst das Start- Feld des Frames auf den Bus und überprüft diesen, indem er vom Bus liest. Da das Startfeld immer ein dominantes Bit ist, sind alle Schreibvorgänge vorerst erfolgreich. Nun beginnen die Knoten bitweise das Arbitrierungsfeld und damit den Identifier auf den Bus zu schreiben und jeweils die geschriebenen Bits zu überprüfen. Liest ein Knoten ein dominantes Bit vom Bus, während er ein rezessives geschrieben hat, bricht er die Arbitrierungsphase ab und geht in den Empfangsmodus über. Es gewinnt der Knoten die Arbitrierung, der den niedrigsten Identifier und damit die höchste Priorität hat. Unter der Voraussetzung, dass ein Identifierwert nur jeweils einer Nachricht zugeordnet wird und dass eine logische Null durch einen dominanten Buspegel abgebildet wird, kann dieser seine vollständige Nachricht senden, während die anderen Knoten auf die nächste Arbitrierung warten müssen.

Für den Fall, dass der Transfer einer Nachricht gleichzeitig von einem Sender als Datentelegramm (Data Frame) und von einem Empfänger als Datenanforderungstelegramm (Remote Frame) initiiert wird, kann der Arbitrierungskonflikt nicht allein über den Identifier gelöst werden. In diesem Fall entscheidet das auf den Identifier folgende RTR-Bit (Remote Transmission Request Bit) über das Buszugriffsrecht. Da mit dem Datentelegramm die von einem Teilnehmer angeforderte Nachricht bereits gesendet

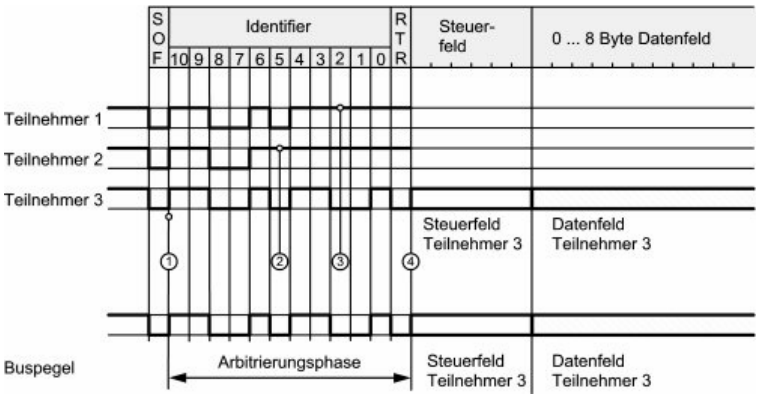

Abb. 3 *Beispiel eines Arbitrierungsvorgangs im CAN-Protokoll. Die Teilnehmer 1, 2, und 3 beginnen gleichzeitig einen Arbitrierungsversuch (1). Teilnehmer 2 verliert zum Zeitpunkt (2), Teilnehmer 1 zum Zeitpunkt (3) das Buszugriffsrecht. Beide Teilnehmer gehen damit in den Empfangszustand. Am Ende der Arbitrierungsphase (4) besitzt nur noch Teilnehmer 3 das Buszugriffsrecht und schaltet seine Nachricht auf den Bus.*

wird, kann die entsprechende Datenanforderung entfallen. Aus diesem Grunde wird das RTR-Bit in einem Datentelegramm mit dem Wert 0, in einem Datenanforderungstelegramm mit dem Wert 1 gesendet.

Die Begrenzung der Länge des Datenfeldes einer CAN–Nachricht auf maximal 8 Byte ist erforderlich für die Realisierung einer möglichst hohen Nachrichtenrate bei kurzer Nachrichtenlänge, um die Latenzzeit für den Buszugang hochpriorer Nachrichten sicher- zustellen, die maximal 130 Bitzeiten beträgt. Außerdem wächst die Wahrscheinlichkeit für das Einfangen von Störungen proportional mit der Blocklänge. Somit ermöglichen die kurzen Nachrichten eine funktionsfähige Datenübertragung auch noch unter sehr schwierigem elektromagnetischem Umfeld.

4 Fehlererkennungs- und Fehlerbehandlungs-Mechanismen

Aus dem ursprünglichen Einsatzgebiet des CAN-Protokolls in Kraftfahrzeugen, forderte hohe Sicherheit auf der Datenübertragungsebene. Um diese zu gewährleisten verfügt das CAN-Protokoll über verschiedene wirksame Mechanismen zur Erkennung verfälschter Nachrichten. Dazu gehören die Verwendung des Cyclic Redundancy Code (CRC), Message-Frame-Check, Acknowledge, Monitoring und des Bit-Stuffing. Die Erholzeit des Gesamtsystems nach einer Fehlererkennung ist kürzer als 30 Bitzeiten (die für das Senden eines Bits benötigte Zeit, abhängig von der Datenrate).

4.1 CRC-Check

Bei der zyklischen Blockprüfung (CRC-Check) wird die zu übertragende Nachricht als Polynom betrachtet und durch ein definiertes Generatorpolynom dividiert. Der Divisionsrest dieser Modulo-2-Division bildet die mit der Nachricht zum Empfänger übertragene Prüfsequenz (CRC-Sequenz). Empfangsseitig wird die empfangene Nach-

richt inklusive Prüfsequenz (Divisionsrest) ebenfalls durch das Generatorpolynom dividiert. Bei fehlerfreier Übertragung ergibt sich kein Divisionsrest.

4.2 Message-Frame-Ckeck

Beim Message-Frame-Check werden die Länge und die Struktur des Frames analysiert. Die fest definierten Formatelemente, wie zum Beispiel rezessive Begrenzungsbits, werden von allen Netzknoten auf Konsistenz überprüft. Wird ein Formfehler (Form Error) detektiert, dann enthielt ein Bitfeld mit festgelegtem Wert ein oder mehrere Bits mit nichtzulässigem Wert.

4.3 ACK-Slot

Ein Sender erwartet im ACK-Slot die Bestätigung des fehlerfreien Empfangs der von ihm gesendeten Nachricht durch mindestens einen Empfänger. Eine fehlende Bestätigung, also kein dominanter Pegel im ACK-Slot, wird vom Sender als nicht angekommene Nachricht und von ihm verursachter Fehler interpretiert.

4.4 Bit-Monitoring

Die Bitüberwachung (Bit Monitoring) durch jeden sendenden Netzknoten überprüft ob der von ihm gesendete Buspegel auch tatsächlich auf dem Bus vorhanden ist. Stimmen die beide Werte nicht überein, so liegt ein „Bitfehler“ vor. Das Überschreiben eines rezessiv ausgegebenen Buspegels durch einen dominanten Pegel wird von einem Sender lediglich während der Arbitrierungsphase sowie während des ACK-Slots akzeptiert. Dadurch werden alle lokale Fehler beim sendenden Knoten und alle global auftretenden Störungen wirksam erkannt.

4.5 Bit-Stuffing

Zur Vermeidung von Synchronisationsproblemen fügt beim Bit-Stuffing der Sender nach fünf aufeinander folgenden, identischen Bits ein komplementäres Bit, das so genannte Stuff-Bit, in den Datenstrom ein. Das Stuff-Bit wird auch dann eingefügt, wenn nach fünf identischen Bits ohnehin ein komplementäres Bit folgen würde. Der Stuff-Bit-Mechanismus gilt nur für Data Frames und Remote Frames. Die Empfänger folgen der gleichen Regel und entfernen die Stuff-Bits aus dem Datenstrom („Destuffing“). Wenn ein Empfänger eine Sequenz von mehr als fünf aufeinander folgenden, identischen Bits erkennt, liegt ein Bit-Stuffing-Fehler vor.

4.6 ”Rest-Wahrscheinlichkeit”

Ein statistisches Maß für die Datenintegrität eines Übertragungssystem ist die „Rest-Wahrscheinlichkeit“, die angibt, mit welcher Wahrscheinlichkeit Nachrichten in nicht erkennbarer Weise verfälscht sein können. Mit diesem Wert kann die Anzahl der wäh-

rend der Betriebszeit eines CAN-Netzwerks nicht erkennbaren Übertragungsfehler abgeschätzt werden.

Ein wichtiges Merkmal des CAN-Übertragungsprotokolls ist die Sicherstellung von netzweiter Datenkonsistenz. Nach dem Aufschalten eines Fehlerflags durch einen Teilnehmer folgt nämlich nicht nur der Abbruch der Übertragung sondern auch das Verwerfen der bereits von anderen Teilnehmern fehlerfrei empfangenen Nachrichten. Außerdem wird durch das sofortige setzen eines Fehlerflags innerhalb eines Bitzeitintervalls (bzw. zwei beim ACK-Slot) die Fehlererkennungszeit gering gehalten. Fehler werden so umgehend erkannt und eine schnelle Wiederholung einer gestörten Nachricht durch den Sender der Nachricht ermöglicht.

Um die Problematik, dass ein defekter Knoten das gesamte System blockieren kann, zu vermeiden, enthält jeder CAN-Knoten einen Sende- und Empfangszähler. Dieser inkrementiert bei fehlerbehaftetem und dekrementiert bei fehlerfreiem Empfangsvorgang. Sie werden jedoch bei Fehlerfällen stärker inkrementiert als sie bei Erfolgsfällen dekrementiert werden. Außerdem findet einen höhere Gewichtung von Fehlern statt, die ein Teilnehmer als erster erkannt und signalisiert hat. Der Stand der Fehlerzähler ist somit ein Maß für die relative Häufigkeit von Störungen und ein Merkmal für wahrscheinlich defekte Knoten. Dieser Mechanismus wird verwendet, um selbständig defekte Netzknoten zu erkennen und gegebenenfalls diese auch automatisch vom Bus abzuschalten.

5 CAN-Scheduler

Man unterscheidet zwischen dem zeitgesteuerten (clock-, time-driven) Scheduler und dem prioritätsgesteuerten (priority- driven) Scheduler.

Zeitgesteuerte Ablaufpläne basieren typischerweise auf dem Master-Slave Mechanismus. Dabei kontrolliert ein Master-Knoten den zeitlichen Ablauf des ganzen Netzwerkes. Alle Prozesse müssen dafür vor Ablauf bekannt sein. Sie werden anhand von Phase, Periode, Ausführungsdauer und Deadline angeordnet und periodisch abgearbeitet. Ein Beispiel für einen solchen zeitgesteuerten Scheduler ist der TT-CAN. Prioritätsgesteuerte Ablaufpläne basieren hingegen auf dem bereits weiter oben erläuterten Arbitrierungs-Mechanismus und sind ereignisgesteuert. Zur Festlegung der Prioritäten sind die Release-Time, die Ausführungsdauer wie auch die jeweilige Deadline ausschlaggebend. Ein Schedule ist eine Zuordnung von Prozessen auf verfügbare Prozessoren.

Die Aufgabe eines zulässigen CAN Schedulers ist die Entscheidung des zeitlichen Ablaufs der Prozesse unter ihrer Prioritäten, die wiederum durch Einhalten von Randbedingungen wie z.B. beschränkte Verfügbarkeit von Ressourcen, Abhängigkeiten zwischen Prozessen und Einhaltung von Deadlines entstehen. Insofern benötigt CAN eigentlich keinen Scheduler. Aber es gibt spezielle auf CAN aufgesetzte Protokolle, die einen Scheduler vorsehen. Sie sind eine Kombination aus ereignisgesteuerten und zeitgesteuerten Ablaufplänen.

5.1 Scheduling Modelle

Periodische Prozesse bestehen aus einem Job, mit festgelegter Ausführungszeit, der innerhalb einer Periode ausgeführt werden muss. Dieses Deadline-Intervall bezeich-

net die Zeit zwischen dem Zeitpunkt, zu dem ein Job ausführbar wird (Freigabezeit) und dem Zeitpunkt an dem er fertig gestellt sein muss. Ein System, dass periodische Prozesse enthält, entspricht einem zeitgesteuertem System (bzw. Scheduler). Sporadische Prozesse haben Jobs mit relativer Deadline, d.h. der Zeitpunkt, an dem der nächste Job gestartet werden muss, ist nicht bekannt, aber der Abstand zwischen zwei Startpunkten ist größer als eine gegebene untere Schranke. Ein System mit sporadischen Prozessen nennt man ereignisgesteuertes System (bzw. Scheduler).

Die Frage, die sich bei Schedulability stellt ist, ob es zu einer Menge von Prozessen einen Ablaufplan gibt, der alle Randbedingungen einhält, wie z.B. bestehende Deadlines. Diese lassen sich durch Frame Response-Time (Frame Antwortzeit) testen.

5.2 WCRT

Die Worst-Case Response Time beschreibt die Situation einer niedrig priorisierten Nachricht, in der alle höher priorisierten Nachrichten zeitgleich senden wollen. Dadurch gelangt die Nachricht in eine Art Warteschleife, in der sie nicht nur auf einen freien Buszugriff, sondern auch nach jeder verlorenen Arbitrierung das Senden der anderen Nachricht abwarten muss, bis sie dran kommt.

Dabei unterscheidet man verschiedene Test-Arten zur Berechnung der Worst-Case Response-Time.

5.3 Der hinreichende und exakte Test

Der hinreichende Test liefert ein wahre Aussage, wenn ein gültiges Schedule existiert. Bei falscher Rückmeldung hingegen kann dennoch mit gewisser Wahrscheinlichkeit ein gültiges Schedule existieren. Der exakte Test besteht aus dem hinreichenden Test und dem notwendigen Test. Dieser liefert falsch, wenn kein gültiges Schedule existiert. Bei wahr hingegen kann dennoch mit gewisser Wahrscheinlichkeit kein gültiges Schedule existieren. Dadurch liefert der exakte Test nie falsche Aussagen über Schedulability.

Für die meisten Anwendungen reicht es aus, die recht pessimistische Response-Time Berechnung des hinreichenden Test zu verwenden. Damit lässt sich die Berechnung der individuelle Frame Response-Time für jede Instanz eines Frames sparen. Die Analyse des exakten Tests ist dagegen dann noch gültig, wenn Deadlines eines Frames größer als ihr Intervall sind.

Der hinreichende Test ist schneller als die meisten Berechnungsformen zur Analyse von schedulability. Größtenteils deklariert er die gleichen Frame-Sets als schedulable wie der exakte Test. Trotzdem bleiben Fälle übrig, in denen der exakte Test als einziger Frame-Sets als schedulable richtig deklariert.

6 Korrekte Response-Time Analyse

Die bis jetzt verwendeten Response-Time Analysen nehmen eine fehlerfreie Kommunikation auf dem CAN-Bus an. Das würde bedeuten, dass alle gesendeten Frames auch korrekt empfangen würden, was in Realität ganz anders aussieht. Störsignale sind gerade im Automobilen Bereich im hohen Maße vorhanden, wie z.B. durch Handys, Radio

und vielem mehr erzeugt. Obwohl diese mittlerweile durch optische Netzwerke umgangen werden könnten, wird dies aus Kostengründen nicht von der Autoindustrie erstrebt. Stattdessen wurde das Problem bei CAN durch die oben bereits erläuterten Fehlerbehandlungs-Mechanismen gelöst.

Das hat wiederum zur Folge, dass z.B. durch das Wiederversenden vorher falsch empfangener Nachrichten die Latenzzeiten eines Frames steigen und mögliche Deadlines nicht mehr eingehalten werden könnten. Ausschlaggebend ist dies bei der Analyse zur schedulability. Um eine korrekte Analyse zu gewährleisten, muss die Berechnung der Worst-Case Response Time um das mögliche Auftreten von Fehlern auf den Bus erweitert werden. Im Laufe der Jahre wurden hierfür diverse Modelle entwickelt. Das erste Modell wurde von Tindell vorgestellt. [*N. Navet, F. Simonot-Lion, Chap. 13-20*]

7 Offsets

7.1 Motivation

Durch die immer weiter ansteigenden Anforderungen an die Bandbreiten eines Bussystems, insbesondere in der Automobilbranche, ist eine Lösung für die Optimierung der Auslastung über den bereits weit verbreiten CAN-Bus, die Verwendung von Offsets. Aber nicht nur die Bandbreite, sondern auch die Aktualität der Daten spielt gerade im Automobilen-Bereich eine große Rolle. Diese hängt unmittelbar mit der WCRT zusammen und wird durch eine geringe maximale Auslastung des CAN-Busses gewährleistet. Durch wachsende Auslastung des CAN-Busses, steigt automatisch auch der WCRT. Der WCRT beschreibt die Situation für ein Frame, bei dem alle höher prio- risierten Nachrichten synchron übertragen werden. Um diese Situation zu vermeiden und damit auch den WCRT zu verringern, wird zunächst die erste Instanz eines Datenstroms eines periodischen Frames zeitverzögert versendet, dem so genannten Offset, vorausgesetzt der Knoten ist zum Senden bereit. Anschließend werden nach dem „Start-Frame“ die darauf folgenden Frames des Datenstroms periodisch versendet. Ist erst einmal das Offset gesetzt, stehen die Sendezeiten der kommenden Frames fest.

Die Wahl des Offsets hat Einfluss auf die Spitzenbelastung des Busses und damit auch auf das WCRT-Verhalten. Um diesen zu verringern werden die frühesten Sendezeitpunkte so weit wie möglich zeitlich gestreut. Dafür wird das Offset jeweils so gewählt dass es möglichst weit von bereits gesendeten Frames ist. Damit sollen synchrone Zugriffe auf den Bus vermieden werden, die wiederum Spitzenlasten bewirken würden. Offsets führen also zu einer Desynchronisation der Datenströme einer Nachricht, was sich wiederum günstig auf die Worst-Case Response-Time auswirkt, da der Datenstrom zeitlich homogener verteilt wird. Durch die Verwendung von Offffsets entstehen Schwierig- keiten, wie zum Beispiel die Wahl des Offsets, die Berechnung der Worst-Case Response-Time, aber auch die Implementierung auf den Knoten.

Zur Berechnung der WCRT wird Software verwendet, wie z.B. Netcarbench, die Datensätze über Netzwerkbelastung, Anzahl der ECUs, die auf die Perioden verteilten Frames und vielem mehr erstellt. [*N. Navet, F. Simonot-Lion*]

Da im CAN-Bus die Nachrichten prioritätsgesteuert versendet werden und nicht zeitgesteuert wie z.B. im TTCAN, wird jede Offset Belegung auf jedem Knoten unabhängig ausgeführt, da keine globale Synchronisation herrscht. Bei der Wahl des Offsets spielen diverse Faktoren eine Rolle. Wie bereits oben erwähnt, arbeitet CAN nach

dem Arbitrierungs-Mechanismus und ist damit ereignisgesteuert. Um für ein ereignisgesteuertes System einen gültigen Scheduler zu finden muss jede Offset Belegung auf jedem Knoten angepasst werden.

7.2 Offsets setzen

Jeder Job wird durch einen Tupel (C, D, T, O) beschrieben. C beschreibt die Worst-Case-Transmission-Time , D die Deadline, T die Transmission-Period und O das Offset. Die Granularität G bestimmt die Zeitabstände zwischen zwei möglichen Release-Times. Die Release-Time R gibt den frühesten Zeitpunkt an, an dem ein Job ausgeführt werden kann. Die Periode ist die Ausführungsdauer, die die Zeit vom Ausführen bis zum Beenden des Jobs beschreibt. Damit ist die Periode auch zeitgleich die Deadline des Jobs, weil der Job sonst mit sich selbst in Konflikt käme. Ein Intervall beschreibt ein Set von aufeinander folgenden möglichen Release-Times.

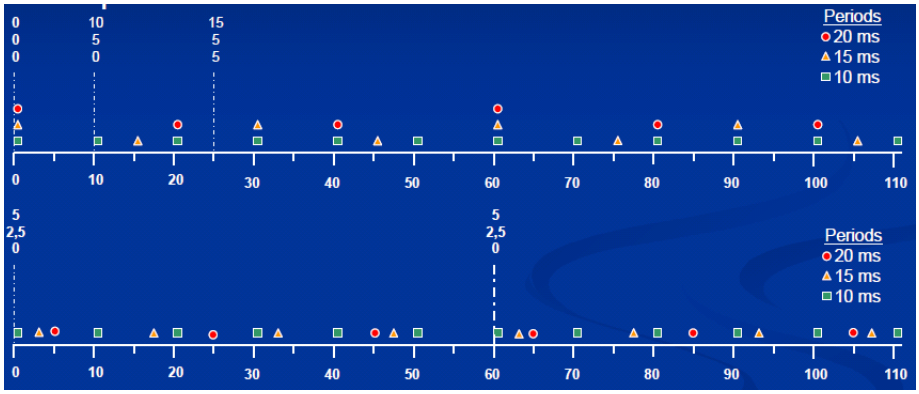

Abb. 4 *Periodischer Zugriff von 3 verschiedenen Jobs mit unterschiedlichenj Deadlines(10ms, 15ms und 20ms). Ohne Offsets(oben) und mit Offsets(unten)*

Zunächst wird angenommen, dass alle Jobs J nach ihrer Periodizität T sortiert sind, sprich diejenigen mit kürzerer zeitlicher Deadline werden als erstes bearbeitet. Der Algorithmus setzt dann iterativ die Offsets von J_1 bis J_n. Betrachtet wird nun ein Durchlauf für J_k, mit 1¡k¡n:
Setze Offset für J_k, so dass die Abstände zwischen den Jobs vor und nach J_k maximiert werden.

1. Suche nach L_k, dem Intervall mit der geringsten Auslastung in $[0, I_k[$, dem Intervall indem alle Jobs mindestens einmal ausgeführt worden sind.
2. Markiere das längste gefundene Intervall L_k mit einem Startpunkt und einem Endpunkt in $[0, I_k[$.
3. Setze das Offset in die Mitte von L_k, was seines Possible Release Time R_k entspricht.
4. Aktualisiere die Release Time, um das gesetzte Offset mit der entsprechenden Release Time im Intervall $[0, I_n[$, zu speichern.

Durch dass gesetzte Offset des ersten Jobs sind alle darauf folgenden periodischen Aufrufe dieses Jobs gekennzeichnet. Weitere Jobs orientieren sich quasi an bereits gesetzte Offsets und finden zwischen den Aufrufen Intervalle, in welchen die geringste Auslastung herrscht.In diesen plazieren sie ihr Offset zentral. Das wiederholt sich bis alle Jobs ihr Offset gesetzt haben. Das Intervall, in dem alle festgelegten Jobs ihre Deadlines erfüllen ist damit schedulable. [*N. Navet, F. Simonot-Lion*]

time	0	2	4	6	8	10	12	14	16	18
possible release time i	1	2	3	4	5	6	7	8	9	10
$R[i]$ (frames released)			$f_{1,1}$		$f_{2,1}$			$f_{1,2}$		$f_{3,1}$

Abb. 5 *f1=(T1=10, O1=4), f2=(T2=20, O1=8), f3(T3=20, O1=18)*

7.3 Ergebnisse

Der hauptsächliche Gewinn durch die Anwendung der Offsets besteht in der Reduzierung der WCRT für niedrig priorisierte Nachrichten. Die Verbesserungen machen sich immer dann deutlicher bemerkbar, je mehr die Priorität abnimmt. In der Praxis sieht es oft so aus, dass ein Großteil der Auslastung ca. 30 Prozent eines Netzwerkes durch einen Knoten verursacht wird. Dieser dient in der Regel als Gateway zwischen den Netz- werken. Erstaunlicherweise ist die ECU Auslastung unter diesen Bedingungen ähnlich zur vorher angenommenen gleichmäßig verteilten.

Um den genauen Vorteil der Nutzung von Offsets ermitteln zu können, fanden Tests statt, die die Auswirkungen auf die Nutzlast mit und ohne Offsets verglichen. Die geplotteten Ergebnisse zeigten deutlich, dass durch die Anwendung von Offsets regelmäßig auftretende Spitzenlasten vermieden werden konnten. Und das nicht nur auf ECU Seite sondern auch bei der CPU, weil dadurch auch das zeitgleiche Senden und Empfangen von Nachrichten vermieden werden kann. Die Konzentration der Auslastung auf einen oder mehreren wenigen Knoten, wie sie in der Praxis oft vorkommt, wurde weiter beobachtet und miteinander verglichen. Es zeigte sich, dass durch die Anwendung von Offsets, die am ausgelasteten Knoten bereits eine 34,5 prozentige Reduzierung der WCRT mit sich brachte und eine 48 prozentige Reduzierung bei den vier meist ausgelasteten auftrat. Um nun auch noch kommende Autogenerationen mit dem CAN-Bus realisieren zu können, schaute man sich an, ob diese signifikanten Verbesserungen auch bei weiterem Anstieg der Datenrate erreichbar blieben. Und tatsächlich konnte festgestellt werden, dass die Performanz, also auch die WCRT, ebenso bei 60 Prozent, also doppelt so hoher Datenrate, noch genau die gleiche ist. Dabei war der Gewinn bei Netzwerken, die die Knotenanzahl beibehielten und die Datenrate erhöhten minimal besser, als bei denjenigen in denen die Knotenanzahl erhöht wurde.

Da aber beim Standard CAN-Bus eine globale Synchronisation vorherrscht, arbeiten die Knoten nach einer lokalen Zeit. Dadurch besteht trotz Desynchronisation durch

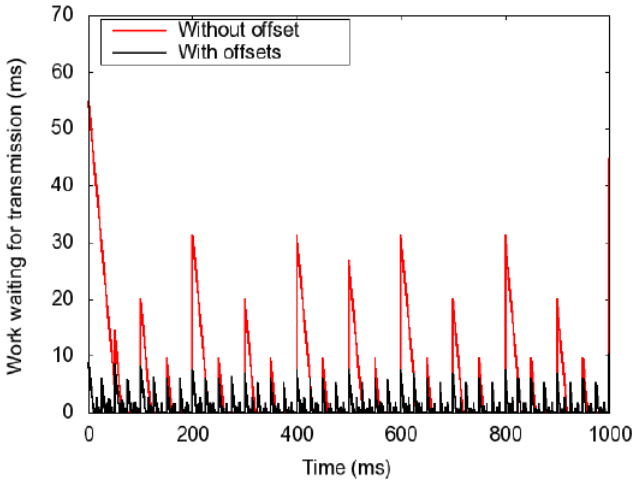

Abb. 6 *Menge an Jobs die auf das Buszugriffsrecht warten, quasi Spitzenlasten bei der Response-Time ohne Offsets und mit Offsets*

die Offsets immer noch die Möglichkeit, dass zwei Knoten zeitgleich auf den Bus zugreifen wollen und jeweils Frames senden wollen. Um dieses Szenario ausschließen zu können, wäre ein neuer Ansatzpunkt zur Optimierung des CAN-Busses nötig. Diese schließt die Synchronisation der ECUs, also eine vereinfachte Ausführung des zeitgesteuerten CAN-Busses mit ein.

8 Fazit

Durch das präsentierte Modell zur Festlegung von Offsets wurde mit geringer Komplexität eine mögliche Optimierung des CAN-Busses vorgestellt. Damit wurde eine Basis geschaffen für weitere Fortschritte und Verbesserungen. Außerdem wurde gezeigt, dass so eine signifikante Performanz Steigerung ermöglicht werden kann und der CAN-Bus dadurch als prinzipielles Netzwerk in den kommenden Auto Generationen weiter verwendet werden kann. Diese Methode kann als Kompromiss zwischen event- und zeitgesteuerter Kommunikation interpretiert werden.

9 Bibliographie

1 . R. I. Davis, A. Burns, Reinder J. Bril and J. J. Lukkien. *Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised.* Real Time Systems, 2007.

2 . J. Goossens and R. Devillers. *The Non-Optimality of the Monotonic Priority Assignments for Hard Real-Time Offset Free Systems.* Real Time Systems, 1997.

3 . J. Goossens. *Scheduling of Offset Free Systems.* Real Time Systems, 2003.

4 . N. Navet and F. Simonot-Lion. *Automotive Embedded Systems Handbook.* Chap. 13/14, CRC Press, Boca Raton, 2008.

5 . K. Etschberger. *CAN Controller-Area-Network: Grundlagen, Protokolle, Bausteine, Anwendungen.* Hanser, München/Wien, 2009.

6 . W. Zimmermann und R. Schmidgall. *Bussysteme in der Fahrzeugtechnik: Protokolle und Standards.* Kap. 3.3, Vieweg und Teubner, Wiesbaden, 2008.

7 . H. Wallentowitz und K. Reif. *Handbuch Kraftfahrzeugelektronik: Grundlagen, Komponenten, Systeme, Anwendungen.* 2006.

8 . B. Heiing und M. Ersoy. *Fahrwerkhandbuch: Grundlagen, Fahrdynamik, Komponenten, Systeme, Mechatronik, Perspektiven.* Vieweg, Wiesbaden, 2007.

9 . O. Alt. *Car Multimedia Systeme Modell-basiert testen mit SysML.* Vieweg und Teubner, Wiesbaden, 2009.

10 Abbildungsverzeichnis

1 . K. Etschberger. *CAN Controller-Area-Network: Grundlagen, Protokolle, Bausteine, Anwendungen.* S. 42, Hanser, München/Wien, 2009.

2 . https://www.vector-worldwide.com/index.php?wbt_ls$_k apitel_id = 450808 root = 376493 seite = vl_einfuehrungcan_de$

3 . K. Etschberger. *CAN Controller-Area-Network: Grundlagen, Protokolle, Bausteine, Anwendungen.* S. 40, Hanser, München/Wien, 2009.

4 . http://www.loria.fr/ nnavet/publi/liste_eng.htm. *Pushing the limits of CAN - scheduling frames with offsets provides a major performance boost.* LORIA - Nancy Université, Lorraine, 2008.

5 . M. Grenier, N. Navet and L. Havet. *Pushing the limits of CAN - scheduling frames with offsets provides a major performance boost.* LORIA - Nancy Université, Lorraine, 2008.

6 . N. Navet and F. Simonot-Lion. *Automotive Embedded Systems Handbook.* Chap.14-11, CRC Press, Boca Raton, 2008.