

Kommunikation in Mobilien Systemen

Seminar Mobile Systeme
SS 2003

Universität Koblenz-Landau
Robert Bärz

ZUSAMMENFASSUNG. In der vorliegenden Arbeit werden Bussysteme aus dem Bereich der Automobiltechnik zur *Kommunikation in Mobilien Systemen* vorgestellt und weiterhin darauf untersucht, wie geeignet sie für zukünftige Technologien (bspw. *X-By-Wire*, *Multimediaanwendungen*, o.ä.) sind. Der CAN Bus, in der Automobilindustrie lange Zeit sehr erfolgreich, wird als Beispiel fuer einen Ereignisgesteuerten Bus erläutert. Neuere Bussysteme verwenden im Unterschied dazu Zeitgesteuerte Netzwerkprotokolle. Hier werden der TTA-Bus und FlexRay als Zeitgesteuerte Busse vorgestellt. Um eine Übersicht über die beiden Bussysteme zu gewinnen, wird ein Vergleich zwischen ihnen im Hinblick auf Leistung, Fehlertoleranz, Flexibilität und Dienstleistungen gezogen.

Inhaltsverzeichnis

1	Einführung	4
1.1	Wozu dient Kommunikation in Mobilen Systemen?	4
1.2	Verbindungsmöglichkeiten von Komponenten	4
1.3	Warum vernetzte Übertragung?	5
1.4	Anforderungen an Busse in Mobilen Systemen	5
2	Der Controller Area Network Bus (CAN Bus)	6
2.1	Meilensteine	6
2.2	Architektur	6
2.3	Datentransfer / Protokoll	7
2.3.1	Sendevorgang	7
2.3.2	Paketaufbau	7
2.4	Grenzen und Probleme	8
3	Zeitgesteuert oder Ereignisgesteuert?	8
3.1	Ereignisgesteuerte Busse	8
3.2	Zeitgesteuerte Busse	8
4	Zwei zeitgesteuerte Busse	9
4.1	TTA Bus	9
4.1.1	Architektur	9
4.1.2	Protokoll	9
4.1.3	Einsatzgebiet	11
4.2	FlexRay	11
4.2.1	Architektur	11
4.2.2	Protokoll	11
4.2.3	Einsatzgebiet	11
5	Vergleich TTA-Bus und FlexRay	12
5.1	Leistung	12
5.1.1	TTP/C	12
5.1.2	FlexRay	13
5.1.3	Fazit	13
5.2	Fehlertoleranz	14
5.2.1	TTP/C	14
5.2.2	FlexRay	14
5.2.3	Fazit	15
5.3	Flexibilität	15
5.3.1	TTP/C	15
5.3.2	FlexRay	15
5.3.3	Fazit	15
5.4	Dienste	15
5.4.1	TTP/C	15
5.4.2	FlexRay	16

<i>ABBILDUNGSVERZEICHNIS</i>	3
------------------------------	---

5.4.3 Fazit	16
-----------------------	----

6 Ausblick	16
-------------------	-----------

Abbildungsverzeichnis

1 Paketaufbau im CAN Protokoll (11 Bit Identifier)	7
2 TTA bus interconnection	9
3 TTA star interconnection	10
4 FlexRay bus interconnection	12
5 FlexRay star interconnection	13

1 Einführung

1.1 Wozu dient Kommunikation in Mobil Systemen?

Im folgenden beschränken wir uns auf die Kommunikation von Geräten innerhalb eines Automobils. Kommunikation bedeutet einen Austausch von Daten. Es gibt verschiedene Stellen an denen Datenaustausch innerhalb eines Automobils stattfindet. Schon heute werden einerseits Daten fuer den Komfort des Fahrers transportiert:

- Scheibenwischer
- Elektrische Fensterheber
- Automatische Lautstärkeregelung des Radios nach Motordrehzahl
- Navigationssystem

und andererseits zwischen Knotenpunkten, die dem Fahrer nicht so bewusst sind:

- Blinker
- Temperatursensoren/Klimaanlage
- Fahrassistenzsysteme wie ABS, ESP

In zukünftigen Generationen von Fahrzeugen wird dieser zweite Teil stark zunehmen. Voraussetzung für Technologien wie *X-By-Wire* (z.B.: Elektronische Lenkung, Autopilot oder Elektronische Schaltung) oder die *Elektronische Deichsel* ist die Kommunikation zwischen einzelnen Komponenten des Fahrzeugs.

1.2 Verbindungsmöglichkeiten von Komponenten

Um eine Kommunikation zwischen einzelnen oder mehreren Komponenten in einem Mobil System zu ermöglichen ist eine physische Verbindung zwischen ihnen notwendig. Denkbar sind zwei Ansätze ([Hyp01]):

- Punkt-zu-Punkt Datenübertragung
- Datenübertragung im Netzwerk

Eine Punkt-zu-Punkt Datenübertragung bedeutet, dass alle Komponenten zwischen denen potentiell Kommunikation stattfinden kann eine direkte Verbindung bspw. durch einen Draht erhalten. Dies hat zur Folge mit der Anzahl der Kommunikationsteilnehmer auch die Komplexität und die benötigte Kabellänge stark anwächst. Im Gegensatz dazu handelt es sich bei der Übertragung im Netzwerk um einen Bus an den die Komponenten angeschlossen sind. Hier hat jeder Teilnehmer die Möglichkeit mit jedem anderen Teilnehmer zu kommunizieren.

1.3 Warum vernetzte Übertragung?

Heute wird in Automobilen im Normalfall das Netzwerk einer Punkt-zu-Punkt Verbindung vorgezogen, weil ein Netzwerk viele Vorteile gegenüber der simplen Punkt-zu-Punkt Verbindung bietet ([Chr02]). Man spricht zum einen von Vorteilen in der Planungsphase:

- Planungssoftware ermöglicht einfache und klare Leitungsführung
⇒ **Zeit und Kosten Ersparnis**
- Alle Daten werden über einen Bus übertragen
⇒ **Einsparung von Kabellängen**
- Installation mit Normbauteilen, automatischer Download der Komponenteneinstellungen
⇒ **Vereinfachte und Schnelle Inbetriebnahme**
- Engineering Tools zur Plausibilitätsprüfung von Geräteeinstellungen
⇒ **Reduzierung von Fehlerquellen**

Zum anderen ergeben sich Vorteile in der Betriebsphase:

- Einsatz von Energiemanagement Applikationen
⇒ **Kostengünstiger Betrieb**
- Vermehrte Abfrage von Prozessdaten
⇒ **Störsicherheit und Zuverlässigkeit**
- Auslesen und vervielfältigen von Geräteeinstellungen
⇒ **Reproduzierbarkeit**
- Vielfältige Diagnosemöglichkeiten
⇒ **Geringer Wartungsaufwand**

1.4 Anforderungen an Busse in Mobilen Systemen

Ein Bus in einem Mobilen System muss gegenüber herkömmlichen Netzwerken besonderen Anforderungen genügen ([DSP]).

Fehlertoleranz:

Fehler einzelner Geräte müssen -auch unter extremen Umständen- verkraftet werden können.

Determinismus:

Festgelegte (niedrige) Latenzzeiten mit minimalen Schwankungen müssen unter Garantie eingehalten werden.

Datendurchsatz:

Eine hohe Datenübertragungsrate muss unterstützt werden.

Fehlererkennung:

Eine schnelle Erkennung (und Anzeige) von Fehlern muss möglich sein.

Einfach, Flexibel:

Einfache Konfigurierbarkeit, Erweiterbarkeit und Flexibilität sind gefordert.

Einheitlich:

Das System muss einheitlich für unterschiedliche Übertragungsmedien (z.B. Glasfaser, Draht) sein.

2 Der Controller Area Network Bus (CAN Bus)

Im folgenden wird der erfolgreiche -aber mittlerweile den Anforderungen nicht mehr genügende- CAN Bus erläutert.

2.1 Meilensteine

Die Entwicklung des CAN Bus (*Controller Area Network Bus*) begann 1983 durch eine Projektgruppe der Firma Bosch. 1986 wurde die erste Version des CAN Protokolls offiziell vorgestellt. Die Markteinführung des Protokolls fand mit der zweiten Version 5 Jahre später statt. Im folgenden Jahr stellte Mercedes die ersten PKW, die zur Kommunikation den CAN Bus nutzen, her. 1993 wird das CAN Protokoll zum ISO Standard (ISO 11898). Ab 1995 nutzen auch weitere Firmen (z.B. Volvo, Saab, Volkswagen, BMW, Renault und Fiat) den CAN Bus in ihren Autos. Um zukünftigen Anforderungen zu entsprechen begann 2000 die Entwicklung einer zeitgesteuerten Erweiterung des Protokolls ([iAC02a]).

2.2 Architektur

Der CAN Bus sieht es vor, dass mehrere Knotenpunkte (*nodes*) in einem Netzwerk zusammengeschlossen werden. Diese *nodes* bilden ein *cluster* in dem Daten priorisiert in einer festgelegten Geschwindigkeit übertragen werden. Möglich sind dabei Datenraten bis zu 1 mbit/sec bei einer Kabellänge von 40m. Wählt man eine niedrigere Datenrate beispielsweise 50 kbit/sec, dann verlängert sich die mögliche Kabellänge auf 1 km.

Es ist möglich mehrere *Cluster* (mit unterschiedlichen Datenübertragungsraten zu einem Netzwerk zu verbinden. Den Übergang vom einem *Cluster* zum anderen bildet dabei ein *gateway-node* ([Kop98]).

2.3 Datentransfer / Protokoll

2.3.1 Sendevorgang

Möchte ein Knoten im Netzwerk eine Nachricht übermitteln, dann erweitert der *can controller* die entsprechenden Daten um die notwendigen Protokoll Daten (*frame*). Danach horcht er den Bus ab bis die aktuelle Übertragung abgeschlossen ist. Jetzt versucht der Knoten via *can controller* die Nachricht zu senden. Hierbei kommt der spezielle Aufbau des *frame* ins Spiel. Am Anfang des *frame* steht ein sogenannter *identifier*, der gleichzeitig den Absender und die Priorität *priority* der Nachricht angibt. Sendet nun ein weiterer Knoten mit einer höheren Priorität gleichzeitig eine Nachricht, so wird der erste Knoten zuerst (weil er eine niedrigere Priorität hat) ein rezensives Bit im Identifier senden. Dieses wird jedoch vom dominanten Bit des höher Priorisierten Knoten überschrieben und sogleich merkt der erste Knoten, der während dem Senden den Bus abhorcht, dass er zur Zeit die Übertragung des höherpriorisierten Teilnehmers nicht stören darf, und unterbricht sofort den Sendevorgang ([iAC02b], [iAa], [iAb]). Dieser Vorgang wird solange wiederholt, bis der Knoten seine Nachricht auf den Bus schicken darf.

2.3.2 Paketaufbau



Abbildung 1: Paketaufbau im CAN Protokoll (11 Bit Identifier)

1. Start of Frame
Markiert den Beginn einer Nachricht
2. Identifier (*Priorität*)
3. Remote Transmission Request
4. Identifier Extension (*Ext. Frame*)
5. Data Length Code
6. Data Field (*bis zu 8 Byte*)
7. Cyclic Redundancy Checksum
8. Acknowledge Bit (*geschrieben von den Empfängern*)
9. End of Frame
10. Interframe Space

2.4 Grenzen und Probleme

Heute ist absehbar, dass der CAN Bus den aktuellen Herausforderungen in Bereich Sicherheit, Fehlertoleranz und der Übertragungsgeschwindigkeit nicht gewachsen ist.

Der Bus kann aufgrund seines ereignisgesteuerten Aufbaus bei hoher Last keine festen Latenzzeiten garantieren. Falls hochpriorisierte Geräte am Bus häufig Nachrichten übermitteln müssen, kann die Verzögerungszeit der niedrig priorisierten Geräte sehr gross werden. Ist die Last am Bus jedoch niedrig, so sind auch die Latenzzeiten der einzelnen Teilnehmer niedrig ([Kop98]).

Die schwankenden Latenzzeiten bereiten auf Applikationsebene grössere Probleme (z.B. passende *timeout* werte zu finden). Der *worst-case* für den CAN Bus ist ein kaputtes hochpriorisiertes Gerät, welches ständig den Bus belegt (*babbling idiot* problem). Dann ist keine Kommunikation auf dem Bus mehr möglich.

Aufgrund der Tatsache, dass alle Teilnehmer gleichzeitig ein Bit senden und empfangen (*transceiver*) können, muss jedes Bit eine gewisse Zeit auf dem Bus erhalten bleiben. Diese Zeitspanne hängt von der physischen Länge des Busses ab. Es ergibt sich also eine durch das Protokoll beschränkte maximale Übertragungsrate je nach Kabellänge. Diese Übertragungsraten reichen für einige Anwendungen nicht mehr aus (z.B. Multimediaanwendungen). Auch geht der Trend heute dahin immer mehr Geräte in einem Automobil durch einen Bus zusammenzuschliessen. Mit der Anzahl der Geräte steigt natürlich auch die benötigte Bandbreite.

3 Zeitgesteuert oder Ereignisgesteuert?

3.1 Ereignisgesteuerte Busse

Das CAN Bus Protokoll ist wie oben beschrieben ein Ereignisgesteuertes Protokoll (*event-triggered*). Ereignisgesteuerte Protokolle kennen keine Globale Zeit. Die Nachrichtenübertragung wird ausschliesslich vom Eintreten bestimmter Ereignisse angetrieben. Dies hat zur Folge, dass eine Nachricht ein **Ereignis** beschreibt, z.B. <Temperaturanstieg auf 25 Grad>, <Bremsvorgang gestart>, <Bremsvorgang beendet> oder <Fensterheber betätigt>. Durch die Ereignisabhängigkeit ist die Netzlast nur schwer vorhersagbar. Auch kann eine Nachricht bei einer hohen Anzahl von Ereignissen in einem kurzen Zeitraum stark verzögert eintreffen ([Kop98]).

3.2 Zeitgesteuerte Busse

Im Gegensatz zu den Ereignisgesteuerten Bussen stehen die Zeitgesteuerten Busse (*time-triggered* oder auch *state-driven*). Hier schickt jedes Gerät am Bus periodisch (in genau festgelegten Zeitscheiben) **Status**-Nachrichten (*state-messages*), z.B. <Temperatur 24 Grad>, <Temperatur 25 Grad>, <Bremskraft 50%>. Sowohl die Netzlast, wie auch die Verzögerungszeit, sind also genau vorhersagbar, da die Nachrichtenübermittlungszeit der einzelnen Teilnehmer von Anfang

an genau festgelegt ist.

4 Zwei zeitgesteuerte Busse

4.1 TTA Bus

Der **TTA Bus** mit dem dazugehörigen Protokoll **TTP/C** wurde von der Technical University of Vienna während der Forschung an Echtzeitsystemen innerhalb von einem Zeitraum von 20 Jahren entwickelt. Der Kommerzielle Vertrieb wird heute von der Firma TTTech (www.tttech.com) unternommen.

4.1.1 Architektur

Der TTA Bus sieht sowohl eine der Abbildung 2 ähnliche Architektur (genannt *TTA bus*), als auch eine Sternförmige Architektur, wie in Abbildung 3 (*TTA star*), vor. In beiden Architekturen ist jeweils ein Host Rechner über ein *network interface* an den Bus angebunden ([Kop02]). Im *TTA bus* benötigt jeder Knotenpunkt einen *bus-guardian*, der die Netzwerkschnittstelle überwacht, so dass kein kaputter Host das gesamte Netzwerk durch ununterbrochenes Nachrichtensenden zerstören kann. Ausserdem überwacht der *bus-guardian* die Übertragungszeiten, damit der Host ausschliesslich innerhalb seiner Zeitscheibe kommuniziert. Dieser *bus-guardian* befindet sich im *TTA star* innerhalb der *star coupler* (siehe Abbildung 3). An dieser Stelle sind also deutliche Kosteneinsparungen, durch nicht mehr pro Host benötigte *bus-guardians* möglich.

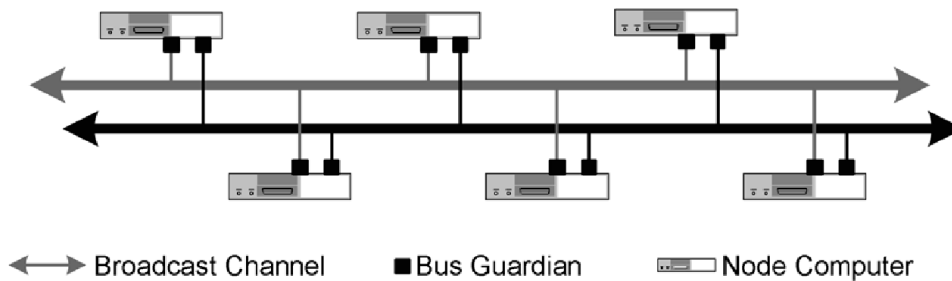


Abbildung 2: TTA bus interconnection

4.1.2 Protokoll

Die Grundvoraussetzung für das Funktionieren des TTP/C Protokolls ist eine für alle Knoten einheitliche Zeit und ein Zeitmodell. Das Übereinstimmen der Uhren (*clock*) wird mittels eines Uhrenabgleichs (*clock-synchronisation*) erreicht. Dieser Uhrenabgleich basiert auf dem Algorithmus von *Welch-Lynch* und toleriert einen defekten Zeitgeber pro Kommunikationsrunde:

Ein Knoten, der seine Uhr abgleichen will, sammelt die Uhrzeiten aller anderen

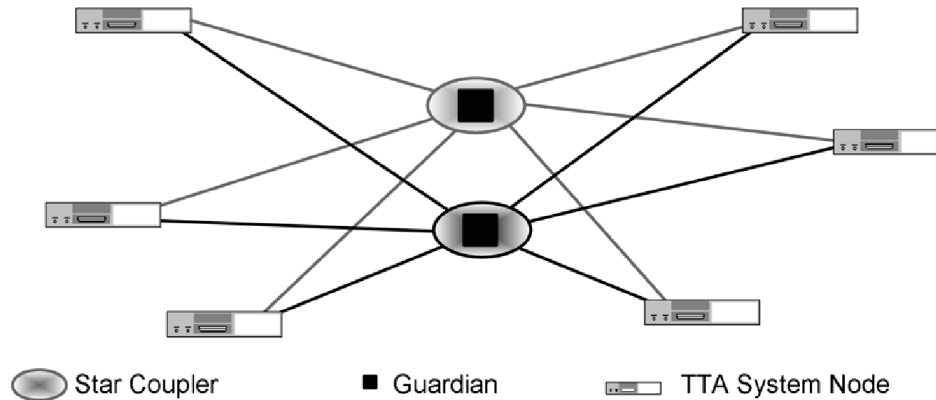


Abbildung 3: TTA star interconnection

Knoten. Seine eigene Uhrzeit berechnet er aus dem Mittelwert des zweitgrößten und des zweitkleinsten Wertes. Um eine Fehlertoleranz von einem defekten Knoten zu erreichen müssen mindestens vier Knoten im Netzwerk sein. Mit weniger Knoten funktioniert der Algorithmus ebenfalls, toleriert jedoch nicht einen fehlerhaften Knoten.

Das Zeitmodell des TTP/C Protokolls teilt die in Wirklichkeit stetige Zeit in diskrete, gleichlange Zeiteinheiten auf. Die Länge dieser Zeiteinheiten hängt von der Genauigkeit der Uhren und des Uhrenabgleichs ab.

Die Kommunikation in TTP/C ist aufgeteilt in einzelne Runden, in denen jeder Knoten innerhalb der ihm zugeteilten Zeitscheibe eine Status-Nachricht absendet. Ist ein Knoten dazu nicht in der Lage, wird er als fehlerhaft angesehen und zu einem Neustart aufgefordert. Bis der Knoten seinen Neustart vollzogen hat, wird er von der Kommunikation ausgeschlossen und trägt seine (evtl. fehlerhafte) Uhrzeit nicht zur *clock-synchronisation* bei.

Eine einzelne Nachricht kann bis zu 240 Datenbytes enthalten, die durch eine 24Bit CRC checksum gesichert sind. Um zu wissen wann er seine Nachricht senden darf, ist im *communication controller* jedes Knoten eine sogenannte *Message Descriptor List (MEDL)* gespeichert. Diese enthält die genaue Zuordnung der Zeitscheiben an die einzelnen Knoten (die Sendeintervalle der einzelnen Knoten dürfen unterschiedlich lang sein um eine bessere Bandbreitennutzung zu gewährleisten). Ein Sender muss also keine Identification übermitteln, sondern wird anhand des Sendezeitpunktes erkannt.

Während jeder Übertragung werden Zeitdifferenzen der Uhren analysiert und bei Bedarf Uhrzeitkorrekturen vorgenommen.

Aufbauend auf diesem Zeitgesteuerten Protokoll kann bei Bedarf ein Ereignisgesteuertes Protokoll (bspw. *TCP/IP* oder *CAN*) aufgesetzt werden. Hierzu

wird in ausgewählten Nachrichten eine bestimmte Anzahl von Bytes für *event-messages* reserviert. Die Übertragung dieser Nachrichten geschieht nach dem *best-effort* Paradigma, sollte also nicht für sicherheitskritische Anwendungen gewählt werden. Durch das Aufsetzen der *event-messages* auf das vorhandene Zeitgesteuerte Protokoll erfahren diese natürlich alle Vorteile, welche auch für *state-messages* vorhanden sind.

4.1.3 Einsatzgebiet

Das TTP/C Protokoll ist gedacht für Sicherheitskritische Anwendungen in Automobilen und in Flugzeugen. Hardware für den Einsatz in einem TTA System ist bereits verfügbar. Auch Softwarelösungen zur Planung und Erstellung von TTA Bus Systemen werden von der Firma TTTech angeboten.

4.2 FlexRay

Vom Industrie Konsortium der Firmen BMW, Daimler Chrysler, Motorola, Philips und anderen wird die Entwicklung des Bussystems **FlexRay** vorangetrieben.

4.2.1 Architektur

Ebenso wie beim TTA gibt es auch für FlexRay zwei verschiedene, kombinierbare Architekturen:

Den passiven Bus und die Aktive Stern Verbindung (Abbildung 4 und 5). Wiederum besitzt jeder Knoten einen Host Rechner, der über eine Netzwerkschnittstelle redundant an den Bus angebunden ist. Bei Flexray befinden sich jedoch auch in der Sternförmigen Architektur die *bus-guardians* in den einzelnen Knoten.

4.2.2 Protokoll

Das FlexRay Protokoll ist ein zweiteiliges Protokoll. Es geht, wie auch das TTP/C, von Kommunikationsrunden aus. Diese sind jedoch in zwei Abschnitte unterteilt: einen statischen Teil für *state-messages* und einen dynamischen Teil für *event-messages*. Es darf einer der beiden Teile fehlen. Die *state-messages* werden von den einzelnen Knoten zu vordefinierten Zeiten abgeschickt. Die *event-messages* werden nach dem Byteflight Protokoll (ein Ereignisgesteuertes Protokoll, entwickelt von BMW) im dynamischen Teil der Kommunikationsrunde gesendet.

4.2.3 Einsatzgebiet

Das FlexRay Protokoll befindet sich zur Zeit noch in Entwicklung. Die FlexRay-Group (www.flexray-group.com) gibt an, dass ab dritten Quartal 2004 erste Hardware, die für der Einsatz in Automobilen qualifiziert ist, bereit sein . FlexRay ist insbesondere als Protokoll für Sicherheitskritische Anwendungen wie

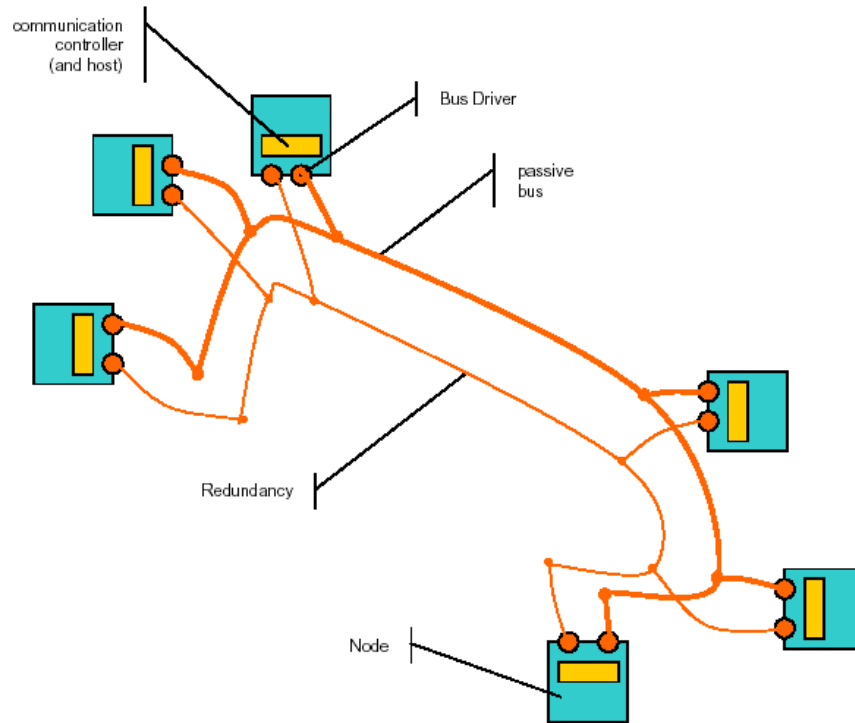


Abbildung 4: FlexRay bus interconnection

X-By-Wire intendiert. Aber auch der Einsatz im Low-Cost-Bereich soll mit einigen Tricks (z.B. der getrennten Nutzung beider Übertragungskanäle) möglich sein.

5 Vergleich TTA-Bus und FlexRay

Im folgenden werden die beiden oben Beschriebenen Bussysteme in den Bereichen Leistung, Fehlertoleranz, Flexibilität und Dienstleistungen miteinander verglichen.

5.1 Leistung

5.1.1 TTP/C

Die seit Ende 2001 verfügbaren *communication-controller* leisten eine Übertragungsrate von 25 MBit/sec. Prinzipiell ist durch das Protokoll die Übertragungsrate nicht beschränkt. Unter Laborbedingungen ist bspw. schon ein System in Betrieb, welches eine Datenübertragungsrate von 1 GBit/sec unterstützt.

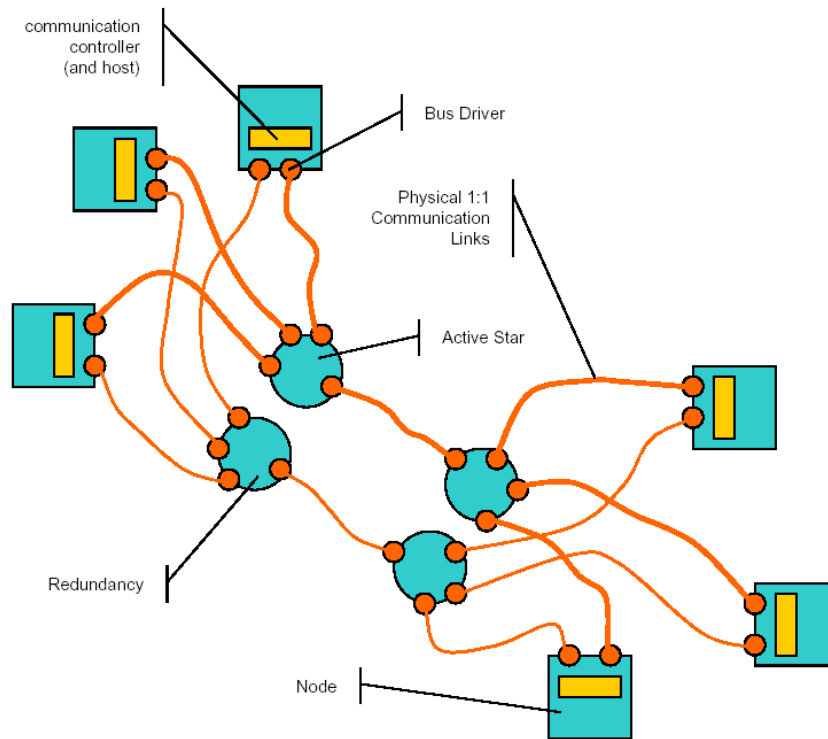


Abbildung 5: FlexRay star interconnection

Das TTP/C Protokoll ist sehr effizient: bei einer Übertragungsrate von 10 MBit/sec wird eine Dateneffizienz von 95,8% erreicht, bei 100 MBit/sec ist die Effizienz noch 78%. Zusätzlich zur hohen Dateneffizienz ist das Protokoll also auch in der Geschwindigkeit gut skalierbar.

5.1.2 FlexRay

Die FlexRay Spezifikation sieht vor, die erste Generation der *communication-controller* mit einer Übertragungsrate von 10 MBit/sec herzustellen.

Das FlexRay Protokoll erreicht bei einer Übertragungsrate von 10 MBit/sec nur eine Dateneffizienz von 45,7%. Bei 100 MBit/sec sind es nur noch 14,5% Effizienz.

5.1.3 Fazit

Das TTP/C Protokoll ist im Bereich der Leistung überlegen, es gibt nicht nur bereits fertige Hardware mit höheren Übertragungsraten, sondern auch die Skalierbarkeit und Dateneffizienz ist dem FlexRay Protokoll deutlich überlegen.

5.2 Fehlertoleranz

5.2.1 TTP/C

Die TTA Architektur erlaubt folgende Fehlermodi:

- (**TTA bus**) willkürliche Störungen *arbitrary faults* im passiven Bus oder den *bus guardians*
- (**TTA star**) willkürliche Störungen *arbitrary faults* im aktiven Hub oder den *communication controllern*
- räumlich begrenzte Störungen *spatial proximity faults* können Knoten und einen Hub zerstören

Maximal beschädigte Elemente:

- (**TTA bus**) In jedem Knoten darf entweder der *bus guardian* oder der *communication controller* defekt sein, ein bus darf defekt sein. Um eine Einzelfehlertoleranz zu erhalten müssen beide Busse funktionieren und mindestens vier Knotenpunkte mit fehlerfreiem *bus guardian* und *communication controller* existieren.
- (**TTA star**) Um eine Einzelfehlertoleranz zu erhalten müssen beide hubs und mindestens 4 Knoten fehlerfrei sein. Ein Betrieb ist auch mit weniger fehlerfreien Komponenten möglich.

Die Fehler dürfen mit einer Rate von maximal ein Fehler alle 2 Kommunikationsrunden auftreten.

5.2.2 FlexRay

FlexRay verkraftet die Fehlermodi:

- willkürliche Störungen in den *communication controllern*
- räumlich begrenzte Störungen *spatial proximity faults* können Knoten und einen Hub zerstören

Maximal beschädigte Elemente:

- in jedem Knoten darf entweder der *bus guardian* oder der *communication controller* defekt sein, ein bus darf defekt sein.
- einer der beiden hubs muss fehlerfrei sein
- für korrekte Uhrensynchronisierung müssen mindestens zwei Drittel der Knoten fehlerfrei sein

5.2.3 Fazit

Die Beschreibung der Fehlertoleranz des FlexRay Busses ist öffentlich nicht so gut zugänglich, wie die des TTA Bus. Die Angaben hier sind daher teilweise Annahmen. Im Gegensatz dazu sind die im TTA Bus verwendeten Strategien sehr gut zugänglich und formal verifiziert. Dies hängt natürlich mit dem Entwicklungsstatus der beiden Projekte zusammen.

5.3 Flexibilität

5.3.1 TTP/C

Die MEDL (*message descriptor list*, also der Zeitplan der Sendezeiten) wird vorausberechnet und ist von Anfang an jedem *communication controller* und jedem *bus guardian* bekannt. Laufzeitänderungen können nur in geringem Masse durchgeführt werden, z.B. die Nachrichtenlänge kann beeinflusst werden. Hier zeigt sich schon, dass die Länge der einzelnen Zeitscheiben für verschiedene Knoten unterschiedlich lang sein kann. Jedem Knoten kann jedoch nur eine Zeitscheibe zugeordnet werden.

5.3.2 FlexRay

Die Zeitplanung im FlexRay ist nicht vorrausberechnet. Jeder Knoten lernt erst beim starten des Busses nach und nach die übrigen Teilnehmer kennen, indem jeder Knoten beim senden auch eine Identifikation übermittelt. Es ist also ein höhere Flexibilität gegeben. Die Länge der einzelnen Zeitscheiben im zeitgesteuerten Teil ist zwar für alle Knoten gleich, ein Knoten kann aber mehrere Zeitscheiben erhalten. Auch die Zweiteilung des Protokolls in einen zeitgesteuerten und einen ereignisgesteuerten Teil erlaubt mehr Variationsmöglichkeiten (siehe oben).

5.3.3 Fazit

Der FlexRay Bus wird seinem Namen gerecht, flexibel zu sein. Allerdings zieht der genutzte Mechanismus, die Übermittlung einer Identifikation in jeder Nachricht, die Nachteile einer schlechteren Dateneffizienz und das Problem der Maskierung mit sich. Es ist nicht klar, wie FlexRay mit einem fehlerhaften Knoten umgeht, der sich als anderer Knoten identifiziert (*masquerading*).

5.4 Dienste

5.4.1 TTP/C

- Uhrensynchronisierung (*clock synchronisation*)
- Nachrichtenkonsistenz (*interactive consistency*)
Alle Übertragungen kommen bei **allen** Empfängern **identisch** an.

- Gruppenzugehörigkeit (*membership service*)
Defekte Knoten können aus dem Bus ausgeschlossen werden, so dass die Übertragungen der funktionierenden Knoten nicht gestört wird.
- *master-shadow configuration*
Ein *shadow node* bekommt den gleichen Sendeslot wie der *master node*. Er darf aber nur senden, falls der *master* ausfällt, andernfalls horcht er nur den bus ab.

5.4.2 FlexRay

Der FlexRay Bus sieht nur Basis Dienste vor. Dazu gehören:

- Uhrensynchronisierung (*clock synchronisation*)
- zuverlässiger Nachrichtentransport (so gut es geht)(*reliable (best effort) message transport*)
Das Protokoll kann nicht garantieren, dass die Nachrichten identisch auf den Empfängern ankommt. Es kann ebenfalls nicht absichern, dass alle Knoten die Nachricht empfangen.

5.4.3 Fazit

Je mehr Dienste auf Ebene des Protokolls unterstützt werden, desto leichter ist das Entwickeln der Anwendungen später. Hier scheint der TTA Bus ausgereifter zu sein. Wiederum sind (vielleicht aufgrund des Entwicklungsstatus) Informationen über Details für den FlexRay Bus nicht verfügbar.

6 Ausblick

Wie Anfangs aufgezeigt wurde, sind die heute eingesetzten Bussysteme den Anforderungen der Zukunft nicht gewachsen. Es muss eine neue Generation eingesetzt werden, die in punkto Sicherheit und Leistungsfähigkeit neue Maßstäbe setzt. Sowohl der FlexRay Bus als auch der TTA Bus haben das Potential dazu. Der TTA Bus ist schon weit fortgeschritten im Entwicklungsstadium und kann in seinem Sicherheitskonzept überzeugen. Auch die Performance des TTA Bus ist im heutigen Vergleich höher als beim FlexRay Bus. Bei FlexRay Bus erhielt während der Entwicklung die Flexibilität eine höhere Priorität im Verhältnis zur Sicherheit als bei seinem Konkurrenten. Im direkten Vergleich ist aus der Sicht des Autors zur Zeit der TTA Bus zu bevorzugen. Allerdings sollte man den FlexRay Bus nicht unterschätzen, da hier die Entwicklung durch ein Konsortium namhafter Firmen stark vorangetrieben wird.

Index

bus-guardian, 9

CAN Bus, 6
can controller, 7
clock, 9
clock-synchronisation, 9
communication controller, 10

Datendurchsatz, 5
Determinismus, 5

Fehlertoleranz, 5
FlexRay, 11
frame, 7

identifier, 7
interactive consistency, 15
interface, 9

Kommunikation, 4

masquerading, 15
master-shadow, 16
membership service, 16
Message descriptor List, 10

Netzwerk, 4

Punkt-zu-Punkt, 4

reliable message transport, 16

star coupler, 9

TTA Bus, 9
TTA bus, 9
TTA star, 9
TTP/C, 9
TTTech, 9

Welch-Lynch-Algorithmus, 9

Literatur

- [Chr02] Dipl.-Ing. Markus Christian. Canopen in der industriehydraulik. Technical report, Bosch Rexroth AG, Industrial Hydraulics, 2002.
- [DSP] Dr. Ralf Schlatterbeck Mag. Markus Novak Dr. Stefan Poledna, Dr. Markus Plankensteiner. Die kommunikationsstruktur für x-by-wire systeme. *Sonderausgabe von ATZ und MTZ und AUTOMOTIVE ENGINEERING PARTNERS*.
- [Hyp01] Automobiltechnologie 2010. Technical report, Hypo Vereinsbank, Mercer Management Consulting, 2001.
- [iAa] CAN in Automation. *CAN Spezifikation 2.0 Part A*.
- [iAb] CAN in Automation. *CAN Spezifikation 2.0 Part B*.
- [iAC02a] CAN in Automation (CiA). Can history. <http://www.can-cia.de/can/protocol/history/history.html>, 2002.
- [iAC02b] CAN in Automation (CiA). Can protocol. <http://www.can-cia.de/can/protocol/>, 2002.
- [Kop98] H. Kopetz. A comparison of can and ttp. Technical report, Technische Universität Wien, Austria, sep 1998.
- [Kop02] H. Kopetz. The time triggered architecture. Technical report, Technische Universität Wien, Austria, oct 2002.