Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Spezifikation von Protokolltransformationen für automotive Anwendungen

Jan Depke
jan.depke@haw-hamburg.de
02.05.2013

Bericht: Projekt 2, Sommersemester 2013
Betreuender Prüfer: Prof. Dr. Franz Korf

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Inhaltsverzeichnis

1 Einleitung

An der Hochschule für Angewandte Wissenschaften (HAW)-Hamburg erforscht die Communication over Realtime Ethernet (Core)-Gruppe zukünftige Automobilnetzwerke auf Basis von Echtzeit-Ethernettechnologien. Die Zielsetzung ist, den schon aktuell steigenden Anforderungen in den Bereichen „Bandbreite“ und „Echtzeitfähigkeit“ gerecht zu werden.

In einem auf dem Realtime Ethernet (RTE) Backbone der core Gruppe der HAW-Hamburg aufsetzenden Automobilbussystem sind Bussysteme wie CAN, LIN oder Flexray über Gateways an ein echtzeitfähiges Paketkommunikationsmedium - das RTE Backbonenetzwerk - angeschlossen. In diesem heterogenen System wird eine Methode zur standardisierten Übertragung von Nachrichten zwischen den unterschiedlichen, angeschlossenen Bussystemen benötigt.

In einem Verbund von Bussystemen existieren unterschiedliche Informationsrepräsentationen, die durch die Eigenschaften und Protokolle der Bussysteme geprägt sind. Wenn Nachrichten die Grenzen eines Bussystems überschreiten, müssen die unterschiedlich strukturierten Nachrichten in unterschiedliche Informationsrepräsentationen transformiert werden. Die Herausforderung ist, eine universelle Methode zur Beschreibung und Durchführung dieser Transformationen zu entwickeln.

Durch die formale Beschreibung der unterschiedlichen Informationsrepräsentationen und der Übersetzungsvorschriften zwischen den Informationsrepräsentationen wird die Grundlage für eine konfigurierbare, algorithmische Transformation einer Busnachricht in ein für Drittbussysteme verständliches Format geschaffen.

Dies ist eine Verbesserung im Vergleich zu heutigen Implementierungen von Kommunikation zwischen unterschiedlichen Bussen im Automobil, welche meist durch zentrale, auf die Kommunikation zwischen nur zwei Bussystemen spezialisierte Gateways gelöst ist.

Diese Ausarbeitung stellt eine Definitionsmöglichkeit von Übersetzungsvorschriften für Nachrichten zwischen unterschiedlichen Bussystemen vor und zeigt deren Handhabung an einem Anwendungsbeispiel.

Der vorgestellte Lösungsansatz wurde dahingehend entwickelt, dass eine Beschreibung der Übersetzungsvorschriften durch Busbeschreibungsstandards wie FIBEX ([ASA11]) möglich ist, dass eine möglichst geringe Auslastung des backbone-Netzwerkes durch Nachrichtenbündelung erreicht werden kann und dass im Regelfall keine Anpassung der Anwendungssoftware der durch die Übertragungsmethode verbundenen Buscontroller notwendig ist.

2 Umgebung

Diese Ausarbeitung wird anhand einer beispielhaften Automobil-Netzwerktopologie erläutert. Die Topologie besteht aus unterschiedlichen Bussystemen wie CAN, LIN oder Flexray, die jeweils über ein Gateway verfügen. Diese Gateways verbinden alle Busse über ein switchgestütztes RTE backbone-Netzwerk (siehe Abbildung 1).

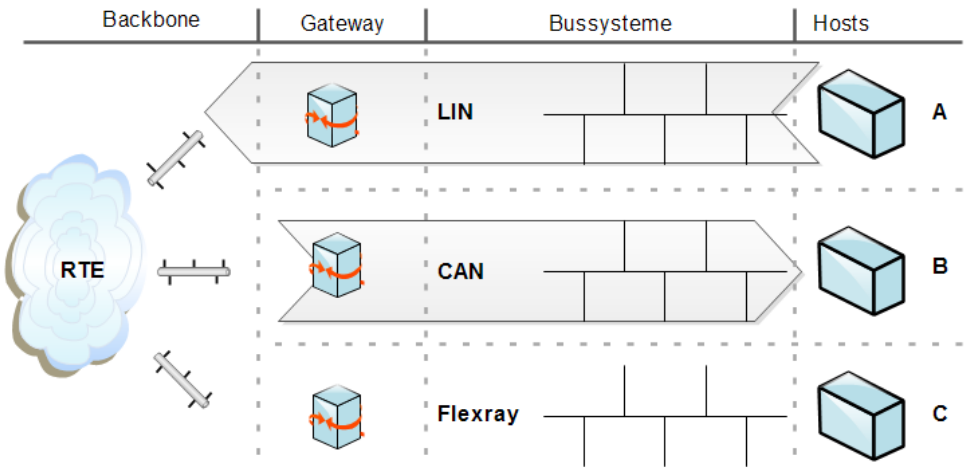

Abbildung 1: Kommunikation zwischen Bussen über ein backbone-Netzwerk

Die Daten einer durch die Pfeile in Abbildung 1 angedeuteten Kommunikation zwischen einem LIN-Busteilnehmer und einem CAN-Busteilnehmer werden an zwei Stellen verarbeitet und in das Protokoll des RTE-backbone bzw des CAN-Bus übersetzt: Im Gateway zwischen LIN-Bus und RTE-backbone und im Gateway zwischen RTE-backbone und CAN-Bus.

Eine Information/Nachricht liegt auf der Transportstrecke in bis zu drei unterschiedlichen Repräsentationen vor, nämlich

- Quelldarstellung,
- Transportdarstellung und
- Zieldarstellung.

Die Betrachtung aus Sicht eines backbone Netzwerkes kann von fremdbestimmten Quell- und Zieldarstellungen ausgehen. Die vorgenannte Verarbeitung innerhalb der Gatewaylogik kann somit als Übersetzung einer Quelldarstellung auf eine Zieldarstellung unter Nutzung einer beliebigen Transportdarstellung angesehen werden. Die Transportdarstellung ist also nicht darauf beschränkt, nur die Daten der Quellnachricht zu beinhalten.

Denkbar ist durchaus, dass mehrere Quellnachrichten innerhalb einer Transportnachricht zusammengefaßt werden, um den durch viele kurze Nachrichten verursachten Protokoll-Overhead zu eliminieren.

Sobald eine Nachricht eines Transportprotokolls analog zu Abbildung 2 mit in einzelne Bitfolgen kodierten Teilnachrichten in einem Gateway eingeht, muss eine jede Teilnachricht mit einer individuellen Übersetzungsvorschrift transformiert werden.

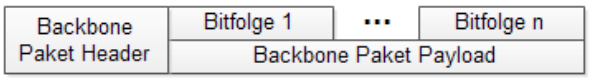

Abbildung 2: Mehrere Nachrichten in einem backbone Frame

Abbildung 2 zeigt mehrere Nachrichten, die über den Mechanismus „Encapsulation“ innerhalb eines Backbone Netzwerkes übertragen werden. Encapsulation bezeichnet eine etablierte Methode in Computernetzwerken, mit der Daten - ggf angereichert um Metainformationen - anhand eines Übertragungsprotokolls transportiert werden können. Die zu transportierenden Daten können hierbei selbst aus mehreren Encapsulationiterationen hervorgegangen sein.

Ein Encapsulationmechanismus bereitet Daten zumeist für die Übertragung zwischen zwei identischen Busprotokollen über ein drittes Busprotokoll auf. In einem solchen Fall kann davon ausgegangen werden, dass aus den Daten des Quellbusses ein semantisch und syntaktisch korrektes Datenpaket des Zielbusses generiert werden kann.

Eine Übertragung unter Beteiligung dreier unterschiedlicher Busse stellt Herausforderungen, die nicht in der Domäne einer Encapsulationmethode liegen. So ist nicht per definitionem gesichert, dass ein Datenpaket eines Quellbusses sich in ein Datenpaket des angestrebten Zielbusprotokolls transformieren läßt. Es wird hier angenommen, dass zur Entwurfszeit sichergestellt wurde, dass die Transformation möglich ist.

Im Folgenden werden Definitionen bzw. Beschreibungsvorschriften zur Konfiguration und Anwendung einer derartigen Encapsulation erarbeitet.

3 Related Work

Die Problematik der Übersetzung zwischen unterschiedlichen Protokollen oder Protokollversionen tritt in vielen Bereichen der Informatik auf.

Eine bezüglich des Verwendungszwecks stark mit dieser Ausarbeitung verwandte Lösung wird in der Avionikindustrie bei der Firma Airbus eingesetzt. Airbus beschreibt die Busschnittstellen und die Übersetzung zwischen diesen Schnittstellen innerhalb eines Flugzeugs anhand eines nicht-öffentlichen „interface control documents“ (ICD) - Standards und stellt Verwaltungssoftware für die Airbus ICD bereit ([nMPDPdlH10]).

Prinzipiell muss zwischen unidirektionalen und bidirektionalen Lösungen unterschieden werden. Unidirektionale Lösungsansätze werden besonders häufig im Kontext von Serviceschnittstellen und Tapping-Devices angewendet. Dort wird vor Entwurf der Übersetzungslogik eines Gateways entschieden, welches Protokoll meist proprietär in ein anderes Protokoll eingebettet werden soll.

Bidirektionale Lösungsansätze - wie zum Beispiel bei der IPv4 / IPv6 Übersetzung - sind wesentlich komplexer zu realisieren. Die Eigenschaften der zu übersetzenden Protokolle erfordern häufig sehr spezielle Lösungen, wie z.B. NAT64 ([BMvB11]) oder NAT-PT ([TS00]). Diese Lösungen müssen häufig eine Adressübersetzung vornehmen, welche aufgrund unterschiedlich großer Addressräume nicht prinzipiell durch ein 1 zu 1 Mapping umsetzbar ist. Ein entscheidendes Merkmal der Mappingvariante ist, ob das Mapping zustandslos oder zustandsbehaftet ist. Ein zustandloses Mapping bedient sich von einer Verwaltungsinstanz fest vorgegebener Addressverknüpfungen zwischen den Protokollen. Ein zustandsbehaftetes Mapping versucht die Addressverknüpfung im Bedarfsfall herzustellen. Das zustandsbehaftete Mapping birgt aber das Risiko, dass zur Laufzeit festgestellt wird, dass eine angeforderte Übersetzung aufgrund einer nicht realisierbaren Addressverknüpfung fehlschlägt.

Es kann im Automobilkontext davon ausgegangen werden, dass die Bussysteme zur Entwurfszeit geplant und getestet werden - insbesondere auch aufgrund der Verwendung von Echtzeitkommunikation und der damit einhergehenden Notwendigkeit, Zeitpläne für eine konkrete Anwendungssituation zu erstellen und zu testen.

In Abgrenzung zu den bekannten Lösungen zur Protokollübersetzung ist diese Ausarbeitung nicht darauf ausgerichtet, eine Zuordnung von Geräteadressen unterschiedlicher Bussysteme und Protokolle herzustellen. Die durch die Ausarbeitung abgedeckten Anwendungsfälle erfordern, dass ein Address- und Nachrichtenmapping vorab der Übersetzungsinstanz bekannt ist und eine zustandslose Übersetzung stattfindet.

Durch die Verlagerung der Verantwortung für die Sicherstellung der prinzipiellen Machbarkeit der Übersetzung an eine dritte Instanz, kann durch diese Ausarbeitung eine algorithmische Übersetzung ohne Beachtung der vorgenannten Hürden beschrieben werden.

4 Lösungsansatz

Die algorithmische Übersetzung einer Quellbusnachricht über ein Transportnachricht in eine Zielbusnachricht erfordert die formale Beschreibbarkeit und Auswertbarkeit dieser drei unterschiedlichen Informationsrepräsentationen. Da die Quellbusnachricht durch das umgebende System bestimmt wird, finden innerhalb einer Übersetzung zwei Transformationen statt: Von der Quellcodierung zur Transportcodierung und von der Transportcodierung zur Zielcodierung. Der in dieser Ausarbeitung beschriebene Lösungsansatz verwendet zwei gleichartig strukturierte Transformation zur Beschreibung einer Übersetzung
Da jede Transformation nur dann algorithmisch umsetzbar ist, wenn die Struktur der zu übersetzenden Nachrichten bekannt ist, muss neben einem Mechanismus zur Übersetzung ebenfalls ein Mechanismus zur Analyse eingehender Nachrichten definiert werden. Die Grafik aus Abbildung 3 stellt den Nachrichtenfluß und die Verarbeitungsschritte einer Übersetzung dar:

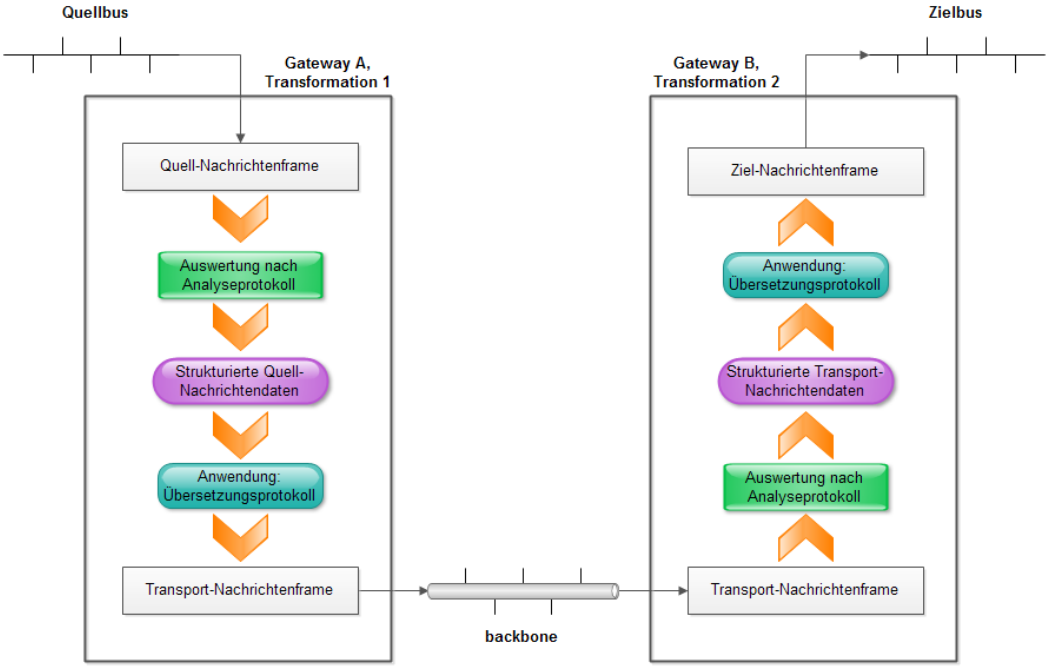

Abbildung 3: Workflow der Übersetzung

Im Sinne von Abbildung 3 wird ein Quell-Nachrichtenframe der Transformationslogik von einer Quellbus-API zur Verfügung gestellt. Ein Analyseprotokoll stellt eine Interpretationsvorschrift für eingehende Nachrichteninhalte dar, die dadurch in eine strukturierte, algorithmisch auswertbare Form gebracht werden. Ein Übersetzungsprotokoll definiert eine selektive Neuordnung von Elementen der strukturierten Informationsdarstellung der Quellnachricht. Ein Übersetzungsprotokoll muss ebenfalls die Möglichkeit bieten, Daten in die Übersetzung einzubringen, die nicht ursprünglicher Bestandteil der Quellnachricht sind. Im Folgenden wird eine formale Beschreibung derartiger Übersetzungsvorgänge definiert.

4.1 Gatewaysicht

In jedem Gateway findet abstrahiert ein Vorgang wie in Abbildung 4 statt. Eingehende Daten eines Quellprotokolls werden erst analyisiert, dann transformiert und letztendlich in ein Zielprotokoll codiert ausgegeben.
Auf diesem Abstraktionsgrad ist es für das Topologiebeispiel nicht bedeutend, ob das Quellprotokoll dieser Sicht nun LIN oder Realtime Ethernet ist bzw, ob ausgangsseitig CAN oder Realtime Ethernet das Zielprotokoll darstellt.
Da ein Gateway mehr Funktionen erfüllt, als die Transformation zwischen unterschiedlichen Informationsdarstellungen, ist eine Betrachtung der Transformation im Kontext der übrigen, relevanten Gatewayfunktionen notwendig. Der folgende Abschnitt beschreibt diese Einordnung.

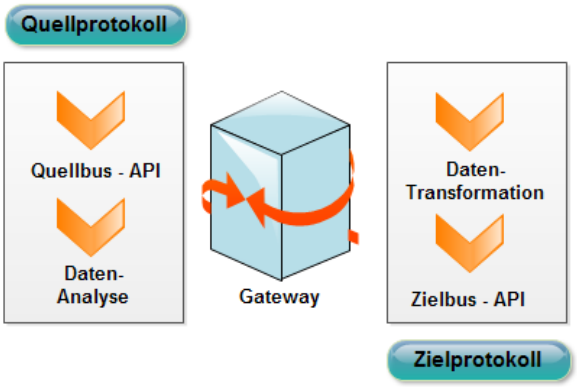

Abbildung 4: Abstrakte Übersetzungsdatenflußsicht einer Gateways

4.2 Einordnung in die Gatewayfunktionen

Eine Transformation in einem Gateway weist noch andere Abhängigkeiten auf, die über die Eigenschaften des Quell- und Zielprotokolls hinausgehen: Gatewayeigenschaften, Konfigurationen, optionale Inhaltsfilterung und Routingentscheidungen beeinflussen eine Transformation. Zusätzlich macht ein Gateway mit Puffereigenschaften, welches mehrere Nachrichten in einem Datenpaket des backbone-Netzwerkes gebündelt übertragen kann, ein Transportprotokoll erforderlich, welches solche Teilnachrichten beschreibt.
Für die Transformation relevante Informationen, die eine Auflösung dieser Abhängigkeiten ermöglichen, werden im Folgenden „Meta-Informationen“ genannt.

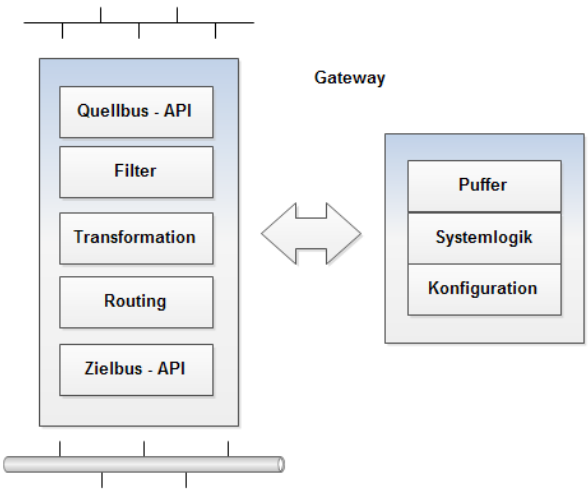

Abbildung 5: Funktionsblöcke eines Gateways

4.3 Funktionsblock 'Transformation'

Unabhängig der Wechselwirkungen einzelner Gatewayfunktionalitäten können die Schnittstellen des Funktionsblocks „Transformation“ (siehe Abbildung 5) wie folgt beschrieben werden: Es erfolgt eine Eingabe einer Bitfolge in einem Quellprotokoll, es erfolgen Ausgaben von Bitfolgen in den Zielprotokollen.

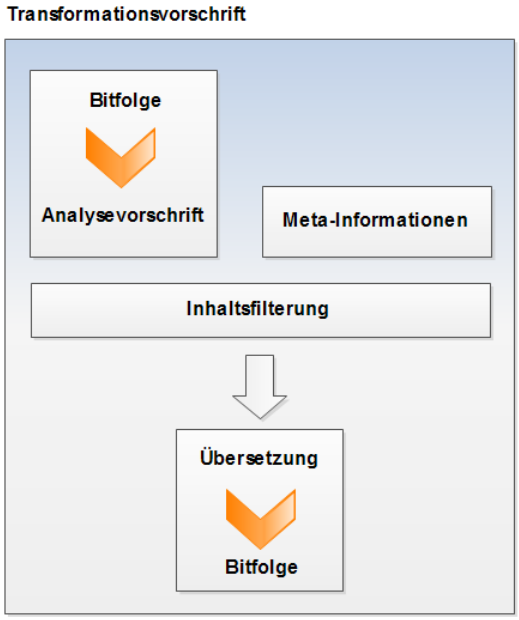

Abbildung 6: Transformationssicht im Gateway

In Abbildung 6 können die Parameter einer Übersetzung identifiziert werden:
1.) Eine Analysevorschrift definiert eine Syntax auf die Eingangsbitfolge, so dass die relevanten Informationen innerhalb der Eingangsbitfolge leicht identifizierbar und referenzierbar sind.
2.) Meta-Informationen aus dem umgebenden System stehen ebenfalls mit bekannter Syntax zur Verfügung.
3.) Ein einfacher Inhaltsabgleich mit relevanten Eingangsinformationen und Meta-Informationen entscheidet über die Anwendung der Übersetzung.
4.) Eine Teilmenge der relevanten Eingangsinformationen und Meta-Informationen wird einer Ausgangsbitfolge mit bekannter Syntax zugewiesen.

Eine darauf aufbauende Betrachtung zeigt den Funktionsblock „Transformation“ aus Abbildung 5 als intelligente Verknüpfung von:

- Analysevorschrift der Quellbitfolge (Analyseprotokoll),
- Metainformationen,
- Konditionalmenge zur Inhaltsfilterung und
- Übersetzungsanweisung (Übersetzungsprotokoll)

4.4 Encapsulationmechanismus

Eine transformierte Quellbusnachricht muss im Payload des Transportnachrichtenframes identifzierbar sein. Dies wird durch einen Nachrichtenheader erreicht, welcher der Quellbusnachricht vorangestellt wird. Dieser Nachrichtenheader unterscheidet sich vom regulären Transportprotokollheader, welcher nur erweiterte Informationen zum Transportprotokoll bereitstellt.

Die im Abschnitt „Einordnung in die Gatewayfunktionen“ genannte Forderung, mehrere transformierte Quellbusnachrichten gesammelt über das backbone-Netzwerk übertragen zu können, bedingt dass jeder Nachrichtenheader einer transformierten Quellbusnachricht innerhalb des Transportprotokollpayloads Informationen zur Nachrichtenlänge und Position enthalten muss. Eine mögliche Anordnung der einzelnen Header und transformierten Quellbusnachrichten innerhalb einer Nachricht des backbone-Netzwerks ist in Abbildung 7 dargestellt.

Abbildung 7: Encapsulation mehrerer Nachrichten im Transportprotokoll

4.5 Formale Definition der Übersetzung

Die algorithmische Übersetzung einer Quellbusnachricht über ein Transportnachricht in eine Zielbusnachricht erfordert die formale Beschreibbarkeit und Auswertbarkeit dieser drei unterschiedlichen Informationsrepräsentationen. Da die Quellbusnachricht durch das umgebende System bestimmt wird, finden innerhalb einer Übersetzung zwei Transformationen statt: Von der Quellcodierung zur Transportcodierung und von der Transportcodierung zur Zielcodierung.
Die Übersetzung selbst bedient sich aus black-box Sicht der Encapsulation. Um eine schärfere Abgrenzung der Begrifflichkeiten „Transformation“ und „Übersetzung“ (Translation) zu erhalten, wird der Übersetzungsvorgang nach seinem Transportmechanismus, der Encapsulation, benannt.

Die formale Beschreibung der Encapsulation wird wie folgt definiert:

4.5.1 Encapsulation

Die Encapsulationmenge faßt Tupel aus Quell- und Zieltransformationssvorschrift zusammen.

Definition 1 (*Encapsulation*)

$$\mathbb{E} = \quad \mathbb{T} \times \mathbb{T}$$

mit $\mathbb{T}$ = *Transformationsvorschrift und*
$\mathbb{E}$ = *Menge der Encapsulations*

Da das kartesische Produkt nicht kommutativ ist, wird die erste Transformationsvorschrift als Kodierung in Richtung des Transportprotokolls und die zweite Transformationsvorschrift als Kodierung Richtung Zielprotokoll definiert.

Die Eigenschaften des kartesischen Produktes erlauben es, einer Transformationsvorschrift in Richtung des Transportprotokolls mehrere Transformationsvorschriften in Richtung des Zielprotokolls zuzuordnen. Dies käme einer Multiplexingfunktionalität im Zielgateway gleich. Im Folgenden werden die Bestandteile einer Transformationsvorschrift vorgestellt

und es werden Möglichkeiten aufgezeigt, die Anwendung einer Transformationsvorschrift mittels durch den Nachrichteninhalt bestimmte Konditionale zu steuern.

4.5.2 Transformation

Eine Transformationvorschrift benötigt ein Analyseprotokoll und ein Übersetzungsprotokoll, um eine Quellbitfolge in eine Zielbitfolge zu transformieren. Die Datenstruktur des Analyseprotokolls ermöglicht die Auswertung des Protokolls eingehender Nachrichten und das Übersetzungsprotokoll definiert, wie das Zielprotokoll dieser Transformationsvorschrift strukturiert ist und mit Nachrichteninhalten gefüllt werden muss.

Sobald das Übersetzungsprotokoll eine Übersetzung hin zum Transport- / backbone-Protokoll beschreibt, kann ein optionaler Transportheader definiert werden. Eine solche Definition ist strukturell identisch zur Definition eines Analyse- oder Übersetzungsprotokolls. Anwendungsbereich eines Transportheaders kann z.B. das Kodieren der benutzen Transformationsvorschrift und der Länge der übersetzten Nachricht sein. Auf diese Art und Weise können diverse übersetzte Nachrichten - durch Hinzufügen für die Auswertung relevanter semantischer Informationen - sequenentiell miteinander verknüpft und gemeinsam übertragen werden.

Definition 2 (*Tranformationsvorschrift*)

$\mathbb{T} = \quad \mathbb{N} \times \mathbb{S} \times \mathbb{S} \times \mathbb{S}$

mit

$\mathbb{N}$ = *Menge der eindeutigen Bezeichner von Transformationsvorschriften,*
$\mathbb{S}$ = *Menge der Protokollfeldsequenzen,*
$\mathbb{T}$ = *Menge der Transformationsvorschriften.*

Ein Tupel $(n, s_1, s_2, s_3) \in \mathbb{T}$ *beschreibt eine Transformationsvorschrift mit*

s_1 = *Analyseprotokollfeldsequenz,*
s_2 = *Übersetzungsprotokollfeldsequenz,*
s_3 = *optionale Protokollfeldsequenz des Transportheaders.*

Der optionale Charakter von s_3 *wird durch die Nutzung eines wirkungslosen Elements* $\in \mathbb{S}$ *erreicht.*

4.5.3 Protokollfeldsequenzen

Ein Protokoll wird durch die Beschreibung der einzelnen Protokollfelder definiert. Im Falle eines eingehenden CAN-Frames wäre z.B. eine Definition über die Protokollfelder „CAN-Nachrichten-ID“ und „CAN-Payload“ möglich. Im Folgenden wird der Begriff Protokollfeldsequenz für die formale Beschreibung der Inhalte und Struktur eines Protokolls verwendet.

Je nach Anwendungsfall kann eine nachrichteninhaltsabhängige Nutzung einer Protokollfeldsequenz in einem Analyse- oder Übersetzungsprotokoll erforderlich sein. Deshalb wird

eine Protokollfeldsequenz hier in Verbindung mit einer Menge optionaler Konditionale definiert, welche Prüfbedingungen auf einzelne Protokollfelder definiert. Nur wenn alle Prüfbedingungen erfüllt sind oder keine Prüfbedingung definiert wurde, wird ein Nachrichteninhalt entsprechend der Protokollfeldsequenz behandelt.

Definition 3 (*Protokollfeldsequenz*)

$$\mathbb{S} = \quad \mathbb{N} \times \{\langle p_i \rangle_{i \in \mathbb{N}, p \in \mathbb{ID}}\} \times \mathbb{COND}^*$$

mit

$b \in \mathbb{N}$ = *eindeutiger Bezeichner eines Protokolls,*
$\mathbb{ID}$ = *Menge der Protokollfeldbeschreibungen,*
$\mathbb{COND}$ = *Menge der Konditionale,*
$\mathbb{S}$ = *Menge der Protokollfeldsequenzen.*

Tupel der Form $(b \in \mathbb{N}, \epsilon, \epsilon)$ *beschreiben besondere Protokollfeldsequenzen, die keine Wirkung als Analyseprotokoll, Übersetzngsprotokoll oder Transportheader haben.*

Die Menge der in jeder Protokollfeldsequenz enthaltenen Protokollfeldbeschreibungen besteht aus einer Folge von Elementen der Menge der Protokollfeldbeschreibungen, wodurch eine Reihenfolge der einzelnen Protokollfeldbeschreibungen etabliert ist.
Eine eingehende Nachricht aus „CAN-ID“ und „CAN-Payload“ muß also durch mindes-

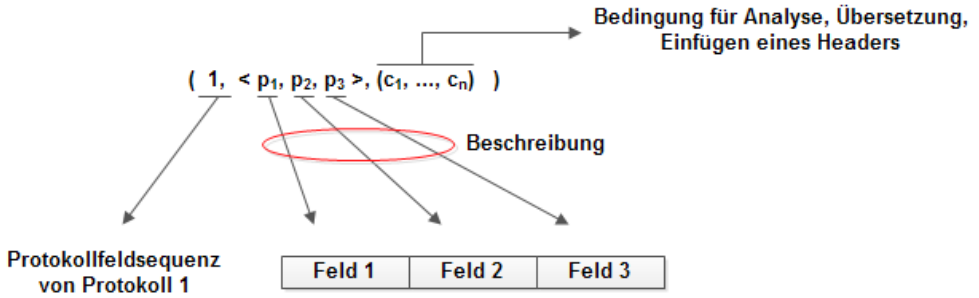

Abbildung 8: Aufbau und Struktur einer Protokollfeldsequenz

tens 2 Protokollfeldbeschreibungen und einen eindeutigen Protokollbezeicher beschrieben werden. Ein passendes Beschreibungstupel ist $(1, p_1, p_2, \epsilon)$, siehe Abbildung 9:

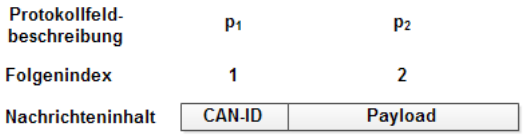

Abbildung 9: Verkürzte Veranschaulichung einer Protokollfeldsequenz (Protokoll 1)

4.5.4 Konditionale

Es kann wünschenswert sein, bei bestimmten Werten innerhalb von Analyseprotokollfeldern eine feiner aufgelöste Auswahl der Transformationsvorschriften durchzuführen. Zum Beispiel kann für CAN-Frames mit den Werten 12 und 15 im Protokollfeld mit dem Folgenindex 2 eine unterschiedliche Transformationsvorschrift gewählt werden. Der in der Transformationsvorschrift aus Abbildung 6 mit „Inhaltsfilterung“ bezeichnete Abschnitt führt Konditionale ein, deren Auswertung als „wahr“/“falsch“ über die Anwendung einer Transformationsvorschrift entscheiden.

Definition 4 (*Konditionale*)

$\mathbb{COND} = \quad \mathbb{N} \times \mathbb{OP} \times \mathbb{OP}_{arg}$

mit

$\mathbb{N}$ = *referenziert eine Protokollfeldbeschreibung* $\in \mathbb{ID}$,
$\mathbb{OP}$ = *Menge der logischen Vergleichsoperatoren,*
$\mathbb{OP}_{arg} \subseteq \mathbb{N}$ = *Menge der Vergleichswerte für logische Operation.*

Konditionale existieren nur in Verbindung mit einer Protokollfeldsequenz. Da Protokollfeldsequenzen eine Folge von Protokollfeldbeschreibungen und somit eine Reihenfolge der Protokollfeldbeschreibungen beinhalten, ist es möglich, ein Protokollfeld über den seine Position / seinen Index in der Folge zu referenzieren.

Um eine Unterscheidung zwischen den Inhalten 12 und 15 im Protokollfeld 2 vorzunehmen, wären z.B. die Tupel $(2, ==, 12) \in \mathbb{COND}$ für $\mathbb{S}_n$ und $(2, ==, 15) \in \mathbb{COND}$ für $\mathbb{S}_m$ geeignet.

4.5.5 Protokollfeldbeschreibungen

Eine Protokollfeldbeschreibung definiert ein einzelnes Protokollfeld eines beliebigen Protokolls. Protokollfelder können je nach Anwendungsszenario durch 5 unterschiedliche Protokollfeldbeschreibungsarten definiert werden:
In Analyseprotokollen können Protokollfelder durch

- eine Bitsequenz statischer Länge,
- eine Bitsequenz dynamischer Länge oder

beschrieben werden. Ein Übersetzungsprotokollfeld kann zusätzlich zur Beschreibung durch eine Bitsequenz durch

- den Rückgabewert einer API-Funktion
- die Referenzierung eines Analyseprotokollfelds oder
- die dynamische Repräsentation eines statischen Analyseprotokollfelds

beschrieben werden. Diese 5 Beschreibungsvarianten werden in der folgenden Definition der Protokollfeldbeschreibungsmenge zusammengefaßt:

Definition 5 (*Protokollfeldbeschreibung*)

$\mathbb{ID} = \mathbb{O} \cup \mathbb{F} \cup \mathbb{N} \cup \{\mathbb{N} \times \{d\} \times \mathbb{N}\}$

mit

$\mathbb{O}$ = *Menge der Bitsequenzteile,*
$\mathbb{F}$ = *Menge der Funktionen,*
$\mathbb{N}$ = *Menge der natürlichen Zahlen,*
$\mathbb{ID}$ = *Menge der Protokollfeldbeschreibungen.*

Die Beschreibung eines Analyseprotokollfelds ist auf Elemente der Menge $\mathbb{O}$ *beschränkt.*

Ein Übersetzungsprotokollfeld, welches durch eine natürliche Zahl beschrieben wird, referenziert ein Analyseprotokollfeld.

Ein Tupel $(n_1 \in \mathbb{N}, d, n_2 \in \mathbb{N})$ *beschreibt ein Übersetzungsprotokollfeld dynamischer Länge durch ein referenziertes Analyseprotokollfeld* n_1 *und die Bitbreite* n_2 *des die Länge der dynamischen Bitsequenz beschreibenden Protokollfeldes.*

Der Bezug zur Protokollfeldsequenz und zur darin enthaltenen Folge von Protokollfeldbeschreibungen eines Analyseprotokolls wird an folgendem Beispiel einer CAN-Nachricht verdeutlicht. Die Nachricht besteht aus insgesamt 35 Bits und wird durch vier Protokollfeldbeschreibungen (hier: Bitsequenzen) eines Analyseprotokolls hinsichtlich ihrer Struktur beschrieben (siehe Abbildung 12):

Die Protokollfeldbeschreibungen eines beispielhaften Übersetzungsprotokolls, welches die

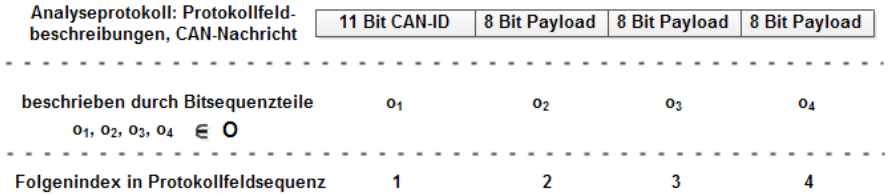

Abbildung 10: Protokollfeldbeschreibungen eines CAN-Analyseprotokolls (Variante 1)

beschriebenen Analyseprotokollfelder verwendet kann wie in Abbildung 11 dargestellt aussehen. Dabei wird die Folge der Protokollfeldbeschreibungen der Protokollfeldsequenz des Übersetzungsprotokolls unter Verwendung einer Funktion, einer Bitsequenz und dreier referenzierter Analyseprotokollfelder beschrieben.

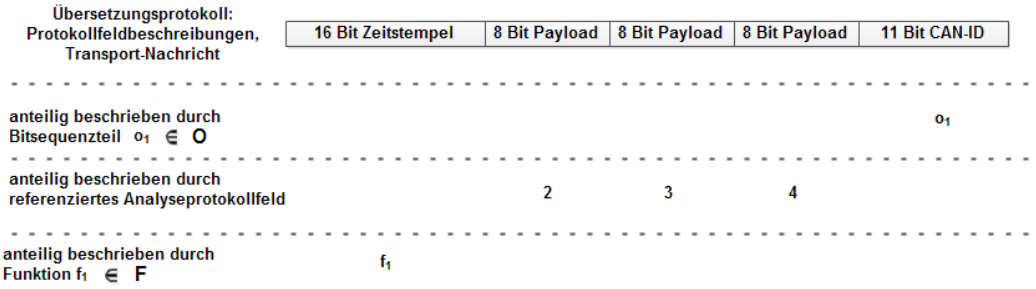

Abbildung 11: Protokollfeldbeschreibungen eines Übersetzungsprotokolls (Variante 1)

Alternativ kann eine identische CAN-Nachricht ebenfalls durch folgende Analyseprotokollvariante beschrieben werden, deren Folge der Protokollfeldbeschreibungen in der Protokollfeldsequenz nur zwei Elemente beinhaltet:

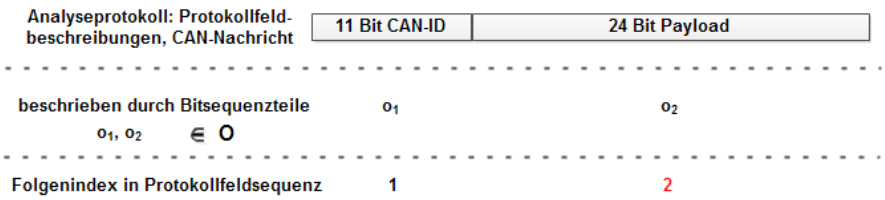

Abbildung 12: Protokollfeldbeschreibungen eines CAN-Analyseprotokolls (Variante 2)

Das im Folgenden vorgestellte, beispielhafte Übersetzungsprotokoll zeigt die Verwendung der dynamischen Repräsentation eines statischen Analyseprotokollfelds (Analyseprotokoll 2, Folgendindex 2). Der Payload des Analyseprotokolls wird hier entsprechend der Protokollfeldbeschreibungsvariante $\mathbb{N} \times \{d\} \times \mathbb{N}$ kodiert (siehe Definition 5).

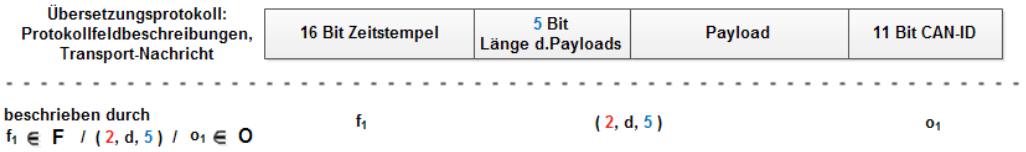

Abbildung 13: Protokollfeldbeschreibungen eines Übersetzungsprotokolls (Variante 2)

Die Menge der Protokollfeldbeschreibungen faßt also die unterschiedlichen Möglichkeiten zur Beschreibung eines Protokollfelds zusammen, die Beschreibungsvarianten durch Elemente der Menge $\mathbb{O}$ (Bitsequenzen) und $\mathbb{F}$ (Funktionen) werden im Folgenden ausgeführt.

4.5.6 Bitsequenzen

Protokolle eingehender und ausgehender Nachrichten setzen sich aus unterschiedlichen Protokollfeldern zusammen, welche innerhalb von Nachrichten einer Bitsequenz entsprechen. Die Bitsequenzen der Protokollfelder ergeben konkateniert die Bitsequenz der Nachricht. Eine Bitsequenz eines Protokollfeldes kann innerhalb der Bitsequenz einer Nachricht durch ihre Position / ihren Offset und ihre Länge identifiziert werden. Da die Protokollfeldbeschreibungen einer Protokollfeldsequenz als Folge definiert sind, existiert eine Reihenfolge der Protokollfeldbeschreibungen. Dadurch kann der Offset eines Protkollfelds bei sequentieller Auswertung der Protokollfeldbeschreibungen berechnet werden und muß nicht angegeben werden .

Es wird angenommen, daß der Übersetzerkomponente eines Gateways die Inhalte der Nachrichten - durch die jeweilige Bus-API vorverarbeitet - als Sequenz statischer oder dynamischer Bitsequenzen übergeben werden. Eine statische Bitsequenz besteht aus einer fixen Anzahl von Bits, eine dynamische Bitsequenz besteht aus einer variierenden Anzahl von Bits. Um die dynamischen Bitsequenzen auswertbar zu gestalten, wird ihnen eine statische Bitsequenz bekannter Länge vorangestellt, deren Inhalt die Länge der dynamischen Bitsequenz beschreibt (Längenfeld einer dynamischen Bitsequenz).

Definition 6 (*Bitsequenzteile*)

$$\mathbb{O}\text{it} = \{\mathbb{N} \times \{0\}\} \cup \{\{0\} \times \mathbb{N}\}$$

$\mathbb{N}$ = *Menge der natürlichen Zahlen,*
$\mathbb{O}$ = *Menge der Bitsequenzteile.*

Ein Tupel $(n \in \mathbb{N}, 0)$ *beschreibt einen Teil einer Bitsequenz durch die Bitbreite des Bitsequenzteils. Ein Tupel* $(0, n \in \mathbb{N})$ *beschreibt hingegen die Bitbreite des der Bitsequenz vorangestellten Längenfelds, aus welchem die Bitbreite des Bitsequenzteils entnommen werden kann.*

Beispiel, statische Bitsequenz:
Ein Tupel (8,0) beschreibt eine statische Bitsequenz der Breite 8 Bit.

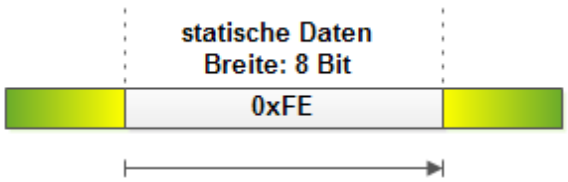

Abbildung 14: Bitsequenz statischer Länge

Beispiel, dynamische Bitsequenz:
Ein Tupel (0,4) beschreibt eine dynamische Bitsequenz variierender Breite. In Kenntnis der Breite (4 Bit) des vorangestellten Längenfeldes, kann die Breite der eigentlichen, dynamischen Bitsequenz ermittelt werden (8 Bit).

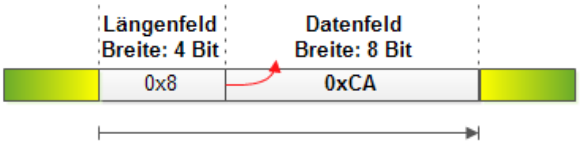

Abbildung 15: Bitsequenz dynamischer Länge mit Längenfeld

4.5.7 Funktionen

Übersetzungsprotokollfeldsequenzen einer Transformationsvorschrift können aber auf Ergebnissen von Berechnungen über Felder der Analyseprotokollsequenz basieren. Ein Anwendungsfall wäre z.B. das Einbetten einer Prüfsumme über Felder der Analyseprotokollfeldsequenz in ein Transportprotokoll. Es müssen also Funktionen definiert werden können, welche ein einzelnes Protokollfeld beschreiben.

Definition 7 (*Funktionen*)

$$\mathbb{F} = \quad \mathbb{FN} \times \{\langle a_i \rangle_{i \in \mathbb{N}, a \in \mathbb{ARG}}\} \times \mathbb{O}$$

mit

$\mathbb{FN}$ = *Menge der Funktionsbezeichner,*
$\mathbb{ARG}$ = *Menge der Funktionsargumente,*
$\mathbb{O}$ = *Menge der Bitsequenzteile, hier: Rückgabewert,*
$\mathbb{F}$ = *Menge der Funktionen.*

Eine Funktion wird hier als 3-Tupel aus einem Funktionsbezeichern, einem Tupel von Argumenten und einer Beschreibung der zurückgegebenen Bitsequenz definiert.

Definition 8 (*Funktionsbezeichner*)

$$\mathbb{FN} = \quad \Sigma^n$$

mit

Σ = *das für Funktionsbezeichner genutzte Alphabet,*
$n \in \mathbb{N}$,
n *sinnvoll groß,*
$\mathbb{FN}$ = *Menge der Funktionsbezeichner.*

Definition 9 (*Funktionsargumente*)

$\mathbb{ARG} = \quad \mathbb{N}^i \cup \mathbb{F}^n$

mit

$i, n \in \mathbb{N}$,
i, n *sinnvoll groß*,
$e \in \mathbb{N}^i$ = *Sequenzposition eines Analyseprotokollfelds*,
$\mathbb{ARG}$ = *Menge der Funktionsargumente.*

Laut Gatewaydefinition gibt es eine Application Programming Interface (API) zur Ermittlung von Werten / Protokollfeldern anhand von ggf. verschachtelten Funktionsaufrufen. Paramter der Funktionsaufrufe sind Indizes von Analyseprotokollfeldern. Die API erlaubt den Zugriff auf unterschiedliche Informationen

- Informationen zum Transport-/Quellbusprotokoll
- Konfiguration der Transport-/Quellbusframeanalyse
- Konfiguration der Übersetzung
- Hilfsfunktionen

und läßt sich in folgende Bereiche unterteilen:

Bus-API:
Es existiert eine Bus-API, deren Funktionen z.B. den Zugriff auf Eingabe- und Ausgabebitfolge an den Schnittstellen der Transformationskomponente des Gateways erlauben oder das Protokoll eines erhaltenen Frames identifizieren. Diese Funktionen sind durch Funktionsaufrufe mit geeigneten Parametern referenziert.

Konfigurations-API:
Es existiert eine Konfigurations-API, welche die aus einer Fibex-Konfiguration gewonnenen Information zur Verfügung stellt. Zu diesen Informationen gehört z.B. die Instanzierung einer Encapsulationdefinition.

Funktions-API:
Es sollen Hilfsfunktionen definiert sein, welche über eine Fibex-Konfiguration referenziert werden können. Diese Hilfsfunktion - wie z.B. ein Prüfsummengenerator - werden entsprechend der Definition von Funktionen beschrieben.

5 Anwendungsbeispiel 1

5.1 Umgebung

Ein LIN-Bus und ein CAN-Bus sind über je ein Gateway und das backbone-Netzwerk verbunden. Im LIN-Netzwerk werden Daten generiert, im CAN-Netzwerk werden Daten konsumiert.

LIN: Im LIN-seitigen Gateway erhält der Translator über die LIN-Buscontroller-API Frames und Informationen zur Bitlänge des Frames.

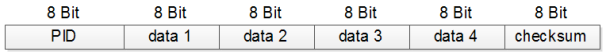

Abbildung 16: LIN Buscontroller API Format

Transport: Der Translator übersetzt die erhaltenen Daten in ein Transportprotokoll und stellt der backbone-Buscontroller-API einen Frame des Transportprotokolls zur Verfügung.

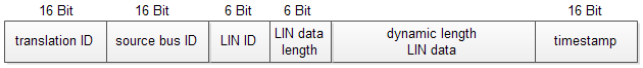

Abbildung 17: backbone Buscontroller API Format

CAN: Im CAN-seitigen Gateway erhält der Translator über die backbone-Bus-controller-API Frames und Informationen zum Frameformat. Der Translator konvertiert die erhaltenen Daten in ein der CAN-Buscontroller-API entsprechendes Format und stellt sie der CAN-Buscontroller-API zur Verfügung.

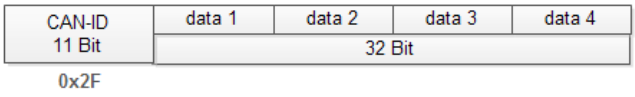

Abbildung 18: CAN Buscontroller API Format 1

Default: Die CAN-Buscontroller-API erwartet für alle LIN-Nachrichten Daten im Format 1 (Abbildung 18) - außer für LIN-Nachrichten mit der ID 12, hier erwartet die CAN-Buscontroller-API angereicherte Daten im Format 2 (Abbildung 19).

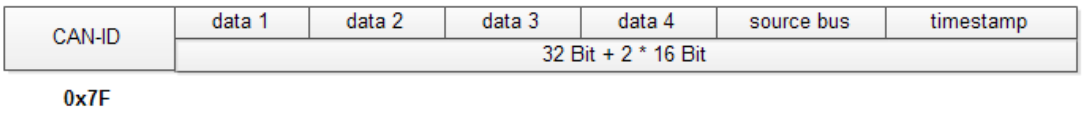

Abbildung 19: CAN Buscontroller API Format 2

5.2 Konfiguration

Die Konfiguration gibt API-Funktionen, beispielhafte Quellbusframes sowie die Instantiierung der Übersetzungsvorschriften und -verknüpfungen vor.

API-Funktionen:

Die 'Funktions-API' stellt dem Encapsulationmechanismus folgende Funktion zur Verfügung:
‘

- timestamp16 - erstellt einen 16-Bit Zeitstempel, parameterlose Funktion
- translationID - gibt den Bezeichner der Transformationsvorschrift zurück
- sourceBusID - gibt den Bezeichner des Quellbusses (LIN) zurück

Die dazugehörigen Funktionen sind entsprechend Definition 7 beschrieben:

$f_1 = (\text{“}translationID\text{“}, \epsilon, (16, 0))$
$f_2 = (\text{“}sourceBusID\text{“}, \epsilon, (16, 0))$
$f_3 = (\text{“}timestamp16\text{“}, \epsilon, (16, 0))$

Somit sind 3 parameterlose Funktionen mit einem 16-Bit langen Rückgabewert definiert. Zusätzlich wird eine Funktion der “Bus-API“ genutzt, über welche das Gateway beim Senden in einen CAN-Bus die zu verwendende CAN-Nachrichten-ID mitteilt:

- getCanID - gibt die vom Gateway ermittelte Ziel-CAN-Nachrichten-ID zurück

Die dazugehörige Funktion ist durch folgendes Tupel beschrieben:

$f_4 = (\text{“}getCanID\text{“}, \epsilon, (11, 0))$

In diesem Fall wird von einer 11-Bit langen CAN-Nachrichten-ID ausgegangen, neuere CAN-Standards mit längeren Nachrichten-IDs lassen sich analog hierzu beschreiben.

LIN-Framedaten:

Es existieren 2 LIN-Nachrichten, für die Übersetzungsregeln definiert werden sollen.

ID: 12 Payload: 0xAA 0xBB 0xCC 0xDD ...
ID: 13 Payload: 0xFF 0xFF 0xFF 0xFF ...

Die LIN-Nachrichten tragen im PID-Byte (siehe 16) ihre 6-Bit ID und 2 Paritätsbits, Nachrichten mit der ID 12 sollen CAN-seitig besonders behandelt werden.

5.3 Transformation in das Transportprotokoll

Im folgenden werden in der Reihenfolge ihrer Nutzung die unterschiedlichen Transformationsvorschriften aufgestellt, welche die Encapsulation beschreiben.

Zuerst wird eine passende Transformationsvorschrift (Definition 2) für die bedingte Transformation in das Transportprotokoll konstruiert. Dazu müssen Analyseprotokoll, Übersetzungsprotokoll, Konditionale und Protokollfeldsequenzen der Transformation hin zum Transportprotokoll definiert werden. Da das Transportprotokoll (siehe Abbildung 17) ein 16-Bit Protokollfeld für einen Zeitstempel trägt, wird zusätzlich eine Definition zur Nutzung der Funktion „timestamp16“ aufgestellt.

Analyseprotokoll:
Das Analyseprotokoll ist eine Protokollfeldsequenz (siehe Definition 3).
Der LIN-Frame aus Abbildung 16 kann ausreichend genau durch folgendes Tupel ($\in \mathbb{S}$) beschrieben werden.

$s_1 = (1, ((6,0),(2,0),(32,0)),\epsilon)$

Durch die Bitsequenzen (6,0) und (2,0) des Analyseprotokolls „1“ wird die PID in den 6 Bit langen ID-Anteil und den 2-Bit langen Paritätsanteil aufgeteilt. Die Bitsequenz (32,0) beschreibt den 4 Byte langen Payload und das Fehlen eines Konditionals wird durch die leere Konditionalmenge $\varnothing$ ausgedrückt.

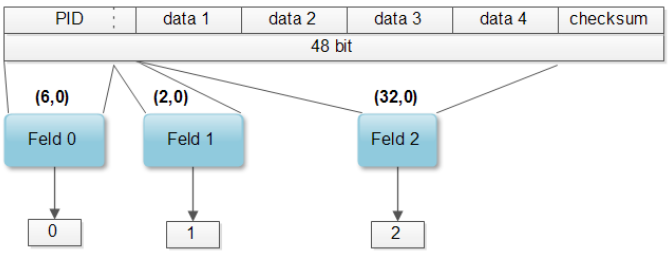

Abbildung 20: Relevante LIN-Protokollfelder

Konditionale:
In diesem Anwendungsbeispiel sollen nur LIN-Nachrichten mit den IDs 12 und 13 übertragen werden. Für diese Unterscheidung werden zwei Konditionale entsprechend Definition 4 beschrieben:

$c_1 = (0, >, 11)$
$c_2 = (0, <, 14)$

Durch den Wert 0 wird das erste Protokollfeld der Analyseprotokollfeldsequenz referenziert, welches die 6-Bit lange LIN-ID beinhaltet.

Die Konditionalemenge $\{c_1,c_2\}$ wird vor der Übersetzung in das Transportprotokoll ausgewertet. Nur LIN-Nachrichten mit der ID 12 oder 13 werden übersetzt (siehe Abbildung 21).

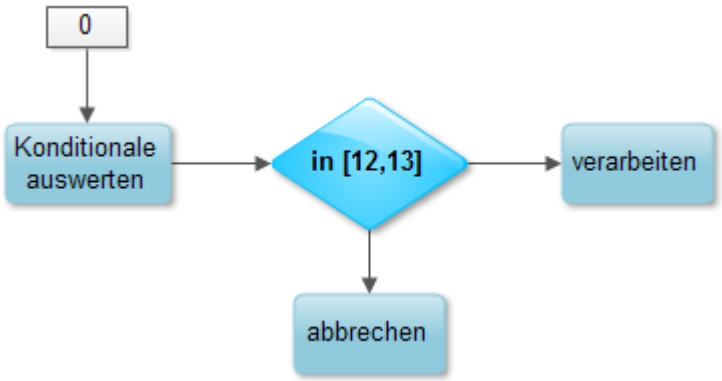

Abbildung 21: Bedingte Übersetzung in das Transportprotokoll

Übersetzungsprotokoll:
Das Übersetzungsprotokoll ist ebenfalls eine Protokollfeldsequenz, nur dass hier auf die Protokollfeldbeschreibung durch Referenzierung eines Analyseprotokollfelds zurückgegriffen werden kann. Entsprechend Abbildung 17 wird das Übersetzungsprotokoll wie folgt beschrieben:

$s_2 = (2, (0, (2, d, 6), f_3), (c_1, c_2))$

Es wird also eine Protokollfeldsequenz bestehehend aus einem statischen Protokollfeld "0" (Kopie des Analyseprotokollfelds 0), einem dynamischen Protokollfeld "2d6" (Analyseprotokollfeld 2, umgewandelt in dynamische Bitsequenz mit 6 Bit Längenfeld) und dem Rückgabewert der Funktion f_3 (statische Bitsequenz, 16 Bit lang) gebildet. Die Transformation wird nur durchgeführt, wenn $c_1 und c_2$ erfüllt sind.

Transportprotokollheader:
Der Transportprotokollheader wird einer transformierten Nachricht vorangestellt. In diesem Anwendungsfall wird entsprechend Abbildung 17 der Transportheader aus "translation ID" und „source bus ID" gebildet, hierzu werden die Funktionen f_1 und f_2 genutzt:

$s_3 = (3, (f_1, f_2), \epsilon)$

Transformationsvorschrift 1:

Alle für die Transformationsvorschrift in Richtung des Transportprotokolls notwendigen Elemente wurden beschrieben, die Transformationsvorschrift entsprechend Definition 2 kann gebildet werden:

$t_1 = (1, s_1, s_2, s_3)$

Die Übersetzung in das Transportprotokoll kann analog zu Abbildung 20 wie folgt visualisiert werden:

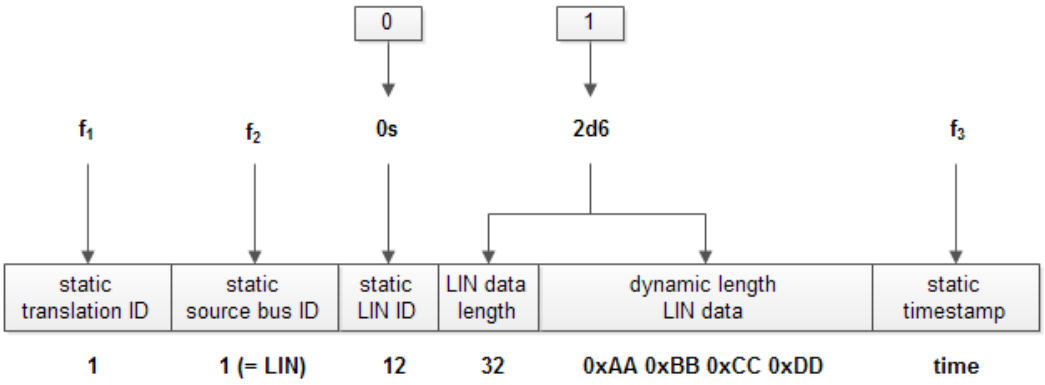

Abbildung 22: Übersetzung in das Transportprotokoll

5.4 Transformation in das Zielprotokoll

Analog zur Aufstellung der Transformationsvorschrift in Richtung des Transportprotokolls kann nun eine Transformationsvorschrift in Richtung des Zielbusprotokolls aufgestellt werden. Es werden erneut ein passendes Analyseprotokoll und Übersetzungsprotokoll benötigt, ebenso wie passende Konditionale für die unterschiedliche Behandlung der ursprünglichen LIN-Nachrichten (siehe Abbildungen 18 und 19).

Analyseprotokoll:

Das Analyseprotokoll wirkt hier auf über das backbone-Netzwerk (RTE) eingehende Nachrichten in Kodierung des Transportprotokolls. Eine geeignete Protokollfeldsequenz zur Auswertung lautet wie folgt:

$s_4 = (4, ((16, 0), (16, 0), (6, 0), (0, 6), (16, 0)), \epsilon)$

Dadurch werden die Nachrichten in die einzelnen Protokollfelder entsprechend Abbildung 22 zerlegt, das Tupel (0,6) bildet unter dem Analyseprotokollfeld 3 die dynamische Bitsequenz aus “LIN data length“ und “LIN data“ ab.

Konditionale:

In Abhängigkeit der LIN-Nachrichten-ID (Analyseprotokollfeld 2) werden unterschiedliche Übersetzungen in das CAN-Protokoll vorgenommen. Dies wird durch folgende Konditionale gesteuert:

$c_3 = (2, ==, 12)$
$c_4 = (2, ! =, 12)$

Übersetzungsprotokoll:

Es werden zwei unterschiedliche Übersetzungsprotokoll benötigt (siehe Abbildungen 18 und 19). Die Default-Übersetzung soll alle LIN-Nachrichten-IDs außer 12 erfassen:

$s_5 = (5, (f_4, 3), (c_4))$

Die Visualisierung der Protokollfeldzuordnung durch die Übersetzung:

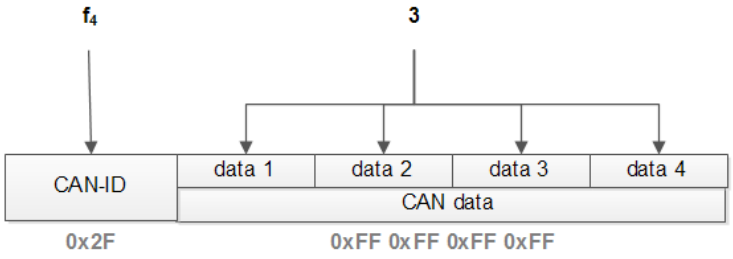

Abbildung 23: Erzeugung eines CAN Frame Typ 1

Sobald eine Auswertung entsprechend des Analyseprotokolls die LIN-Nachrichten-ID 12 ergibt, wird ein besonderes Übersetzungsprotokoll verwendet, welches neben des LIN-Payloads ebenso die Quellbus-ID und den Zeitstempel in den CAN-Payload einfügt:

$s_6 = (6, (f_4, 3, 1, 4), (c_3))$

Die entsprechende Visualisierung der Protokollfeldzuordnung durch die Übersetzung:

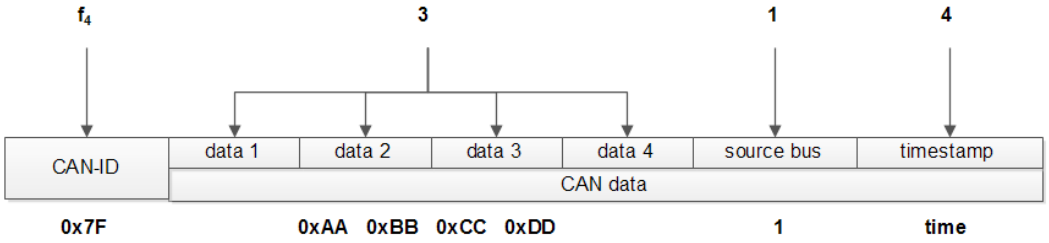

Abbildung 24: Erzeugung eines CAN Frame Typ 2

Um die Verwendung eines wirkungslosen/leeren Transportheaders zu demonstrieren, wird die durch die Funktion f_4 ermittelte CAN-Nachrichten-ID direkt in die Übersetzungsprotokolle eingebunden. Alernativ hätte für die CAN-Nachrichten-ID ein eigener Transportheader definiert werden können.

Transportprotokollheader:

Ein Transportprotokollheader mit einer leeren Menge von Protokollfeldbeschreibungen und einer leeren Konditionalmenge soll dazu führen, dass der Transportprotokollheader bei der Übersetzung ignoriert wird.

$s_7 = (7, \epsilon, \epsilon)$

Transformationsvorschriften 2 und 3:

Aus den beschriebenen Elementen lassen sich zwei Transformationvorschriften bilden, deren Anwendung durch die Konditionale c_3 und c_4 gesteuert wird.
$t_2 = (2, s_4, s_5, s_7)$
$t_3 = (3, s_4, s_6, s_7)$

5.5 Encapsulation

Die drei aufgestellten Transformationsvorschriften können nun zu Elementen der Menge der Encapsulations zusammengefaßt werden.

$e_1 = (t_1, t_2)$
$e_2 = (t_1, t_3)$

6 Anhang

Literatur

[ASA11] ASAM. Asam mcd-2 net 4.0.0, fibex. http://www.asam.net/nc/home/standards/standard-detail.html?tx_rbwbmasamstandards_pi1[showUid]=1246&start=, 2011.

[BMvB11] M. Bagnulo, P. Matthews, and I. van Beijnum. Stateful nat64: Network address and protocol translation from ipv6 clients to ipv4 servers, 2011. RFC 6146.

[nMPDPdlH10] Miguel Ángel Mozas Pajares, Carlos Murciano Díaz, Ismael Lafoz Pastor, and Carlos Fernández de la Hoz. Icd management (icdm) tool for embedded systems on aircrafts. http://web1.see.asso.fr/erts2010/Site/0ANDGY78/Fichier/PAPIERS%20ERTS%202010%202/ERTS2010_0151_final.pdf, 2010. Visit: 27.04.2013.

[TS00] G. Tsirtsis and P. Srisuresh. Network address translation - protocol translation (nat-pt), 2000. RFC 2766.

Abbildungsverzeichnis

Abkürzungsverzeichnis

API Application Programming Interface

Core Communication over Realtime Ethernet

RTE Realtime Ethernet

HAW Hochschule für Angewandte Wissenschaften

Glossar

'bit stream' / 'Bitfolge':
Ein bit stream sei eine endliche Folge von Werten $\in \{0, 1\}$.

'statisches Protokollfeld':
Ein Protokollfeld sei eine Teilfolge eines bit stream mit bekannter Position Länge „length“, welcher eine semantische Bedeutung zugeordnet werden kann. Seine Position ist nur in Zusammenhang mit einer Protokolldefinition errechenbar.

'dynamisches Protokollfeld':
Ein Protokollfeld sei eine Teilfolge eines bit stream mit dynamischer Länge „length“, welcher eine semantische Bedeutung zugeordnet werden kann. Die Länge des Protokollfeldes wird hierbei durch den Wert ein vorangestellten, statischen Protokollfeldes definiert.

'statisches Protokoll':
Ein statisches Protokoll SP sei eine Folge von statischen Protokollfeldern, wobei die Anzahl und Ordnung der Protokollfelder unveränderlich definiert sind.

'dynamisches Protokoll':
Ein dynamisches Protokoll SP sei eine Folge von statischen und dynamischen Protokollfeldern.

'Protokolldefinition':
Eine Protokolldefinition definiert die notwendigen Protokollfelder eines bestimmten Protokolls. Aus der Protokollfelddefinition ergibt sich somit ebenso die Ordnung der Protokollfelder.

'Protokolle':
Die Menge $\mathbb{P}$ sei die Menge aller im Sinne der Problemstellung interessanten Protokolle.

'Frame':
Ein Frame sei ein bit stream, der einer Protokolldefinition folgend sinnvoll interpretiert werden kann.