OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

INFORMATIK UND
MATHEMATIK

Bachelorarbeit

Erweiterung eines Netzwerkkontrollers zur automatischen Konfigurierung, der sich im Fahrzeug befindenden Netzwerkgeräte, anhand verschiedener Applikationsparameter

Eingereicht von:	Brunthaler Sebastian
	Matrikelnummer: 3032075
	Studiengang: Allgemeine Informatik
Arbeitgeber:	Continental Automotive GmbH
	Siemensstr. 12, D-93055 Regensburg
	Betreuer: Christian Humig
Hochschule:	Ostbayerische Technische Hochschule Regensburg
	Erstprüfer: Prof. Dr. Thomas Waas
	Zweitprüfer: Prof. Dr. Markus Kucera

Inhaltsverzeichnis

Tabellenverzeichnis

Sperrvermerk

In dieser Bachelorarbeit sind Geschäftsgeheimnisse der Continental Automotive GmbH enthalten. Eine Weitergabe oder Vervielfältigung oder Veröffentlichung der Arbeit sowie die Verwertung und Mitteilung ihres Inhaltes ist ohne ausdrückliche schriftliche Genehmigung der Continental GmbH, Siemensstraße 12, 93055 Regensburg nicht gestattet.

Glossar

AVB Audio Video Bridging

CAN Controller Area Network

CBS Credit Based Shaper

ECU Electronic Control Unit

ID Identifikationsnummer

IEEE Institute of Electrical and Electronics Engineers

ISO International Organization for Standardization

JSON JavaScript Object Notation

LIN Local Interconnect Network

MAC Media Access Control

MACsec Media Access Control Security

OSI Open Systems Interconnection

PTP Precision Time Protocol

QoS Quality of Service

REST Representational State Transfer

SDN Software-defined networking

TAS Time Aware Shaper

TSN Time-Sensitive Networking

XML Extensible Markup Language

Kapitel

1

Einleitung

1.1 Motivation der Arbeit

Die Mobilitätssysteme stehen heute vor vielfältigen Herausforderungen: Globalisierung und Urbanisierung lassen das Verkehrsaufkommen rapide anwachsen und könnten Verkehrssysteme an ihre Kapazitätsgrenzen bringen. So werden im Jahr 2050 schon 70 Prozent aller Menschen in Städten leben. Die Zahl der Automobile wird sich verdoppeln. Automatisierung und Vernetzung bieten jedoch die Chance, diese globalen Herausforderungen erfolgreich zu bewältigen, weil Autofahren damit effizienter, sicherer und umweltverträglicher wird. Die deutschen Hersteller und Zulieferer wollen ihre Innovationsführerschaft beim automatisierten und vernetzten Fahren weiter ausbauen.

Auch die Bundesregierung hat in ihrer Strategie zum automatisierten und vernetzten Fahren das Ziel ausgegeben, Deutschlands Vorreiterrolle bei dieser Technologie zu sichern. Das automatisierte und vernetzte Fahren soll auf die Straße gebracht werden: über den Probebetrieb und die Entwicklung zur Serienreife bis zur Regelzulassung. Das Bundesministerium für Verkehr und digitale Infrastruktur hat dafür folgende Handlungsfelder definiert, in denen die nötigen Voraussetzungen für die neue Technologie geschaffen werden sollen: Infrastruktur, Recht, Innovation, Vernetzung sowie IT-Sicherheit und Datenschutz.

Die Anzahl der Fahrerassistenzsysteme, die den Fahrer bei der Fahraufgabe unterstützen können, hat in den letzten Jahren weiter zugenommen. Dadurch ist der Eindruck entstanden, dass autonome Fahrzeuge schon bald technisch machbar sein könnten. Viel wahrscheinlicher ist es jedoch, dass diese Entwicklung evolutionär verlaufen wird. Automatisierte Funktionen werden auf der Grundlage etablierter Fahrerassistenzsysteme sukzessive weiterentwickelt und in mehr und mehr Neuwagen eingebaut. Autofahrer werden so Schritt für Schritt an die Automatisierung herangeführt.

Dabei wird automatisiertes Fahren voraussichtlich zuerst auf der Autobahn sowie in Parkhäusern praktisch angewendet werden. Auf Bundesautobahnen ist trotz hoher Geschwin-

digkeiten das Verkehrsgeschehen vergleichsweise strukturiert. In Parkhäusern hingegen herrschen geringe Geschwindigkeiten vor, die die Situation trotz hoher Komplexität beherrschbar machen. In beiden Fällen kann ein Auto sein Umfeld mit den eigenen Sensoren erfassen, die Situationen sind damit gut beherrschbar. Es liegt auf der Hand, dass weitere Innovationen erprobt werden müssen, so zum Beispiel die Ausweitung der Anwendung automatisierter Funktionen im urbanen Umfeld. Die Bundesregierung hat daher erste Initiativen ergriffen, zum Beispiel das Digitale Testfeld Autobahn, das um städtische Testfelder erweitert werden soll. [1]

1.2 Projekt A3F

Anhand von Schlüsselszenarien sollen die Herausforderungen beim Übergang hin zu einer neuen flexiblen und dynamischen Fahrzeugarchitektur herausgearbeitet werden. Hierbei geht es darum, die Anforderungen auf Applikationsebene zu verstehen, um daraus dann die Technologien und Lösungskonzepte für das Netzwerk im Fahrzeug der Zukunft abzuleiten. Ziel ist es, unter dem Gesichtspunkt der Wiederverwendung vorhandener Methoden und Technologien aus der Enterprise IT zu beleuchten und diese hinsichtlich eines möglichen Einsatzes im Fahrzeug zu bewerten. Hierzu müssen diese Methoden und Technologien neben den funktionalen Eigenschaften auch noch in Hinblick auf die speziellen automotive Anforderungen betrachtet werden. Darüber hinaus sollen erste Ansätze zu einem möglichen Migrationspfad, von bestehenden zu zukünftigen, an automotiven Anforderungen angepasste Lösungen aufgezeigt werden. Am Ende des Projekts sollen die gewonnen Ergebnisse anhand eines Demonstrators gezeigt werden. Daraus lassen sich folgende 3 globale Ziele definieren:

- Identifizierung von Schlüsseltechnologien und Methoden für zukünftige Fahrzeug-Architekturen
- Daraus Empfehlungen für zukünftige Kommunikationsnetzwerke im Fahrzeug ableiten
- Transfer von potentiellen IT-Technologien ins Fahrzeug unter Berücksichtigung von automotive Anforderungen

1.3 Aktuelle Problemstellung im Fahrzeugnetz

Jede neue Anforderung an ein Fahrzeug hat aktuell eine Anpassung des Fahrzeugnetzwerks zur Folge. Dies will man in Zukunft vermeiden, indem das Netzwerk so dimensioniert wird, dass es durch beliebige Anwendungen erweitert werden kann. Außerdem muss es mit der Zunahme an Datenmengen zurecht kommen, da beispielsweise das autonome Fahren einen enormen Zuwachs an Kommunikationsdaten benötigt.

Mit den gestiegenen Anforderungen an Bandbreite, Zeiteinhaltung und Skalierbarkeit ist das aktuelle Fahrzeugnetz nicht mehr ausreichend dimensioniert und muss für diese Zwecke überarbeitet werden.

1.4 Ziel dieser Arbeit

Wie bereits erwähnt ist das Ziel ein Fahrzeugnetz zu entwickeln, welches durch funktionelle Applikationen erweitert werden kann. Hierzu sollte das Konzept von Plug and Play verwendet werden, d.h. verschiedene Anwendungen müssen aus einem zentralen Anwendungsverzeichnis geladen werden können und ohne manuelles Eingreifen in das Fahrzeugnetz lauffähig sein.

Die automatische Konfiguration soll durch ein Applikationsmanifest ermöglicht werden. Es gibt die Randbedingungen an, welche zum reibungslosen Betrieb der Anwendung benötigt werden. Das Manifest soll bereits vom Entwickler der Anwendung, anhand der spezifischen Anforderung an das Netzwerk im Fahrzeug, festgelegt werden. Da alle Anwendungen nicht die selben Netzwerkparameter benötigen, ist es wichtig bereits bei der Entwicklung an die nötigten Eckdaten zu denken. Basierend auf diesen Parametern, wird eine bereits bestehende zentrale Netzwerkkontrolleinheit, der sog. Netzwerkmanager, funktionell weiterentwickelt. Dieser soll in Zusammenarbeit mit weiteren Komponenten im System eine automatische Konfiguration des Netzwerkes vornehmen.

Die Veranschaulichung des Anwendungsfalles der Arbeit ist in Abb.1 dargestellt. Diese zeigt den Netzwerkmanager, hier zur Vereinfachung als Blackbox dargestellt. Anhand der Eingabeparameter aus dem Manifest allein können noch keine Netzwerkkonfigurationen erstellt werden. Der Manager muss auch über die Eigenschaften der Netzwerkteilnehmer Bescheid wissen. Nicht jedes Gerät kann die geforderten Bedingungen mit den dafür erforderlichen Technologien umsetzen. Aus diesem Grund sendet jede beteiligte Komponente seine Eigenschaften ebenfalls an den Netzwerkmanager. Dieser wertet sein Input automatisch aus und erstellt eine Konfiguration für alle Teilnehmer der Kommunikationsstrecke im Fahrzeugnetz.

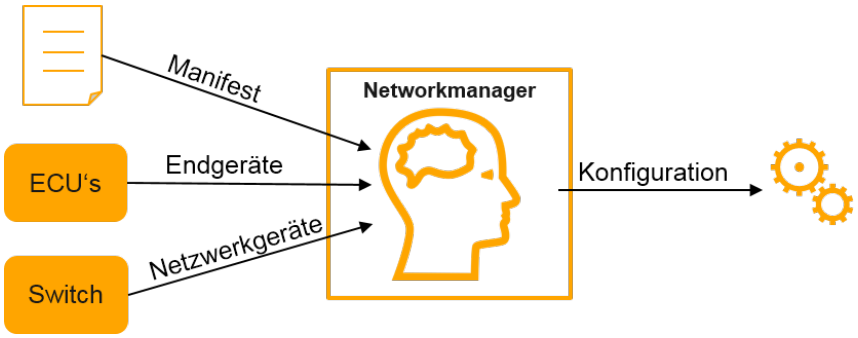

Abbildung 1: Vereinfachte Darstellung der Netzwerkkonfiguration

Im Laufe der Arbeit werden Endgeräte als Electronic Control Unit (ECU) bezeichnet. Netzwerkgeräte sind Switche und andere vermittelnde Komponenten im System.

Die Ausarbeitung des Themas erfolgt nicht nur theoretisch, sondern wird teilweise praktisch umgesetzt. Um dies zu ermöglichen müssen mehrere verschiedene Features implementiert werden. Der Praktische Teil beschränkt sich jedoch nur auf ein Fallbeispiel, da dies sonst den zeitlichen Rahmen der Arbeit sprengen würde. Anhand des Testfalles soll die Funktionalität der automatischen Konfiguration mittels eines Applikationsmanifestes belegt werden.

Kapitel

2

Grundlagen zu Echtzeitsystemen und Netzwerken in Fahrzeugen

Dieses Kapitel beschreibt die Grundlagen im Bereich Netzwerktechnik für Fahrzeuge. Im Punkt Echtzeitsysteme stützt sich diese Arbeit auf die Normen der Projektgruppe IEEE 802. Die Analyse dieser Arbeit wird auf den definierten Standardisierungen basieren. Dies umfasst die Definition der verwendeten Begriffe und die Erklärung von Ethernet-Grundlagen. Im Anschluss erfolgt die Erläuterung von Echtzeitsystemen auf Basis der TSN-Protokolle.

2.1 Definition und Begriffe

Im folgenden Unterkapitel werden Begriffe und Definitionen erläutert, welche in den weiteren Kapiteln relevant sind. Es wird auf deterministische Echtzeitfähigkeit, TSN im Hinblick auf den Automobil-Bereich und QoS Parameter mit deren Einflüsse und Ethernet in Fahrzeugen eingegangen.

2.1.1 Deterministische Echtzeitfähigkeit

Bei den zunehmend komplexeren Aufgaben eines Kommunikationssystems in einem Fahrzeug wird es immer schwieriger die Echtzeitfähigkeit eines Systems zu garantieren. Daten die mit der Anforderung an Echtzeit versendet werden, bezeichnet man auch als kritische/zeitkritische Daten. Diese müssen innerhalb einer vorgegebenen maximalen Zeitverzögerung beim Empfänger zur Verfügung sehen. Durch die in Unterabschnitt 2.1.2 definierten QoS-Parameter, können Garantien für die Verfügbarkeit von kritischen Daten getroffen werden.

Die Haupteigenschaften eines deterministischen Ethernet-Systems sind:

- Vorhersagbarkeit von Ereignissen
- Berechenbarkeit des Systems
- Konsistente Antwortzeiten zwischen den Endverbrauchern

Der Netzwerk-Determinismus ermöglicht es, Daten garantiert in einem definierten Zeitfenster zwischen Endsystemen auszutauschen. Um eine deterministische Übertragung sicherzustellen, muss die Beeinflussung durch nicht planbare und nicht vorhersagbare Ereignisse im Netzwerk verhindert werden. [2]

2.1.2 QoS-Parameter

In aktuellen Netzwerken werden verschiedene Dienste angeboten. Diese können durch ihre benötigten Parameter definiert werden. Die Parametrisierung von Diensten ist im Standard ISO/IEC 13236 als Quality of Service (QoS) definiert und bezeichnet die Dienstgüte. Die vorhandenen Netzwerkdienste können anhand ihrer Güte spezifiziert und klassifiziert werden. Das Prinzip von QoS wird im Werk Computer Networks von A. Tannenbaum anschaulich erläutert [28]. Im Wesentlichen ergeben sich folgende vier Hauptparameter, Latenz, Jitter, Paketverlust und Bandbreite. Im weiteren Verlauf werden diese erläutert.

2.1.3 Latenz

Die Latenz beschreibt in erster Linie die Performance eines Systems. In dieser Arbeit wird hauptsächlich auf die Latenzzeit beim Übermitteln von Nachrichten eingegangen. Diese beschreibt die Zeit, die zwischen der Versendung und dem Empfang eines Datenpaketes im Netzwerk benötigt wird. Durch anderen Datenverkehr im System kann es zusätzlich zur Erhöhung der Latenz kommen. Der sog. Querverkehr kann zum Problem werden, falls nicht-kritische und kritische Daten gemeinsam über eine Kommunikationsstrecke fliesen. Querverkehr (eng. *interspersing traffic*) bezeichnet in dieser Arbeit Daten, die den eigentlichen Datenfluss beeinträchtigen. Hier kann es bei jedem Hop zu Verzögerungen kommen. Ein Hop bezeichnet einen Netzwerkknoten in einem System. [26] [23]

Die Abbildung 2 zeigt ein Beispiel zur Versendung von Nachrichtenpaketen $p_1 \dots p_6$ von zweier Anwendungen *A* und *B* über einen Switch (ECU). Hier kommt es an verschiedenen Stellen zu Verzögerungen der Pakete. Diese bezeichnet man als Delays. Ein Netzwerkswitch besitzt mindestens eine Warteschlange (*Queue*).Die Daten werden anhand der Media Access Control (MAC)-Adresse an die angegebene Zieladresse weitergeleitet. In diesem Beispiel hätte das Paket p_1 die MAC-Adresse von *C*.

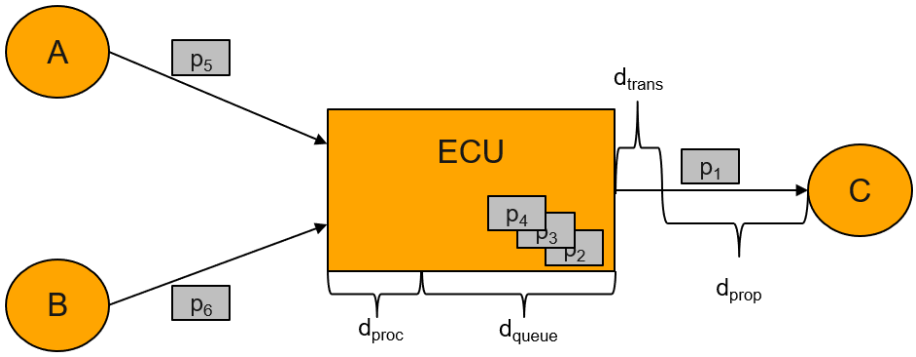

Abbildung 2: Entstehung der Latenzzeiten bei Übermittlung von Datenpaketen

In den folgenden Punkten wird auf die verschieden Delay-Arten eingegangen:

- Verarbeitungsverzögerung (Processing Delay d_{proc})
  Beim Empfangen des Datenpakets am Switch wird es anhand der MAC-Adresse zum entsprechenden Ausgangsport weitergeleitet. Die Verzögerung entsteht bei der Verarbeitung des Paket-Headers. Hier wird neben der Zieladresse auch auf mögliche Fehler, wie beispielsweise Bit- und Übertragungsfehler, geprüft. Mit steigenden Anforderungen an die Kommunikation, können hier auch weitere Prüfungen stattfinden [27, S. 1629-1634]. Die Verarbeitungszeitverzögerung ist auch abhängig von der Implementation des Switches. [23]

- Warteschlangenverzögerung (Queuing Delay d_{queue})
  Jeder Port zum Datenversand hat eine Möglichkeit zur Datenzwischenspeicherung mittels verschiedener Queues. Diese ist nötig, da immer nur nacheinander Daten übertragen werden können. Ist ein Port gerade belegt, wird das Paket in die zugehörige Queue eingereiht. Die Wartezeitverzögerung berechnet sich aus der Summe von noch zu versendenden Paketen. Je mehr Pakete bereits in einer Warteschlange sind, desto höher ist die Wartezeit. Ein Port kann auch priorisierte Warteschlangen haben. Hierbei hat er mehrere Queues, die je nach Wichtigkeit der Datenpakete befüllt werden und nach diesem Prinzip abgearbeitet werden. [23]

- Übertragungsverzögerung (Transmission Delay d_{trans})
  Die Übertragungsverzögerung bezeichnet die Zeit, die benötigt wird um ein Datenpaket komplett auf die Leitung zu legen. Sie steht somit in direkter Abhängigkeit zu der Paketgröße. [23]

- Ausbreitungsverzögerung (Propagation Delay d_{prop}) Die Weiterleitung zum nächsten Switch oder Endpunkt der Nachricht wird durch die Ausbreitungsverzögerung angegeben. Diese ist direkt proportional zur Leitungsgeschwindigkeit, welche wiederum von der physikalischen Beschaffenheit des Mediums abhängt. [23]

Alle diese Delay-Zeiten aufsummiert, ergeben die Latenzzeit einer Nachricht. Befinden sich auf der Kommunikationsstrecke mehrere Switche, den Hops, so müssen diese Verzögerungen auch mit beachtet werden.

2.1.4 Jitter

Mit *Jitter* wird die Schwankung der Verzögerungszeit bezeichnet und gibt die maximale Differenz zwischen Ende-zu-Ende-Verzögerung an. Die Hauptgründe für Schwankungen sind vor allem Abweichungen beim Zugriff auf Übertragungsmedien und Querverkehr im Datenstrom. Jitter kann sowohl zu Verspätungen als auch zum verfrühten Ankommen von Datenpaketen führen. Durch diese Schwankungen sind Laufzeitgarantien weniger genau möglich. In der Netzwerktechnik wird Jitter auch oft als Varianz oder Laufzeit von Datenpaketen bezeichnet. [26] [23]

2.1.5 Paketverlust

Es gibt verschiedene Arten von Paketverlust. Die häufigsten Verluste sind Fehler bei der Übertragung und Verluste bei der Verarbeitung in Switchen. Bei Letzterem kommt es bei zu vielen Datenpaketen zu einem Überlauf einer Queue. Dies bezeichnet man als *Buffer-Overflow*. Hierbei gehen die zuletzt eingegangen Pakete verloren, da sie nicht mehr zur Verarbeitung in der Warteschlange platziert werden können. Ein Überlauf kommt nur dann zustande, wenn die Größe der eingehenden Daten höher ist als die durchschnittliche Datenrate mit der die Pakete aus den zugehörigen Queues versendet werden können. [26] [23]

2.2 Ethernet Grundlagen

In den folgenden Unterkapiteln wird kurz auf Ethernet spezifische Grundlagen eingegangen, welche im Laufe dieser Arbeit relevant werden. Es wird keine vollständige Definition von Ethernet erfolgen. Diese kann bei Bedarf in den zusätzlich angegeben Quellen recherchiert werden. Hier wird lediglich auf die Grundlagen und die für die Arbeit relevanten Eigenschaften des Ethernet eingegangen.

2.2.1 Ethernet Frame

In Abbildung 3 wird ein kompletter Ethernetframe auf Layer 2 des ISO/OSI-Modell grafisch dargestellt. Das ISO/OSI-Modell ist ein standardisiertes Referenzmodell für Netzwerkprotokolle. Es ist in sieben verschiedenen Schichten gegliedert. Dies kann detailliert unter Kapitel 1.4 im Werk von A. Tanenbaum nachgelesen werden [23].
Ein Ethernet Paket ist in folgende Blöcke unterteilt:

- MAC-Adressen:
  - Destination Address: Gibt die Zieladresse des Paketes an
  - Source Address: Gibt die Adresse des Versenders an
- Ethertype: Gibt den Typen des Kommunikationsprotokolls an [5]
- Payload: In diesem Block sind die zu versenden Daten
- CRC Checksum: Prüfung des übertragenen Frames auf Fehler, wie beispielsweise Bit-Fehler

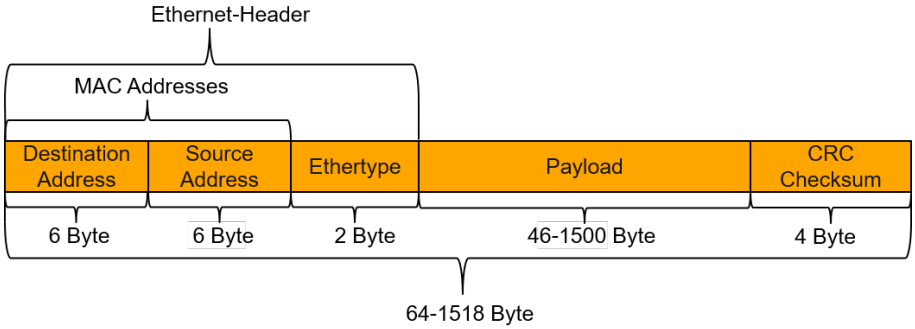

Abbildung 3: Ethernet Frame auf Layer 2 Ebene

Die ersten drei Blöcke bilden zusammen den Ethernet-Header eines Pakets. Dieser wird vom weiterleitenden Switch oder Empfänger als erstes gelesen.

2.2.2 Media Access Control Security (MACsec)

MACsec ist ein Prinzip, das die Sicherheit einer Punkt-zu-Punkt-Ethernet-Verbindung ermöglicht. Es handelt sich um einen Mechanismus, der sich auf Layer 2 im ISO/OSI Modell befindet. Wenn MACsec verwendet wird, muss der Ethernetframe abgeändert werden.
Grundsätzlich werden in einer Ende-zu-Ende Kommunikation die Sicherheitsschlüssel der beiden Teilnehmer überprüft. Nur wenn diese übereinstimmen wird die Kommunikation erlaubt. Wenn MACsec aktiv ist, wird nicht nur die Kommunikation selbst geschützt, sondern es findet auch eine Integritätsprüfung der gesendeten Daten durch das Feldes *Integrity Check Value* statt. Diese enthält eine Prüfsumme, die über die geschützten Daten gebildet wird und sich ändert, falls diese verfälscht werden. Die Überprüfung der generellen Verbindung wird anhand der Daten im Header des Frames durchgeführt. Besonders wichtig für das Prinzip von MACsec ist hierbei der sog. *SecTAG* [16], welcher in Abb. 4 dargestellt wird. Dieser Bereich unterteilt sich nochmals in fünf Blöcke, anhand derer verschiedene Informationen und Sicherheitsmechanismen definiert sind [16].

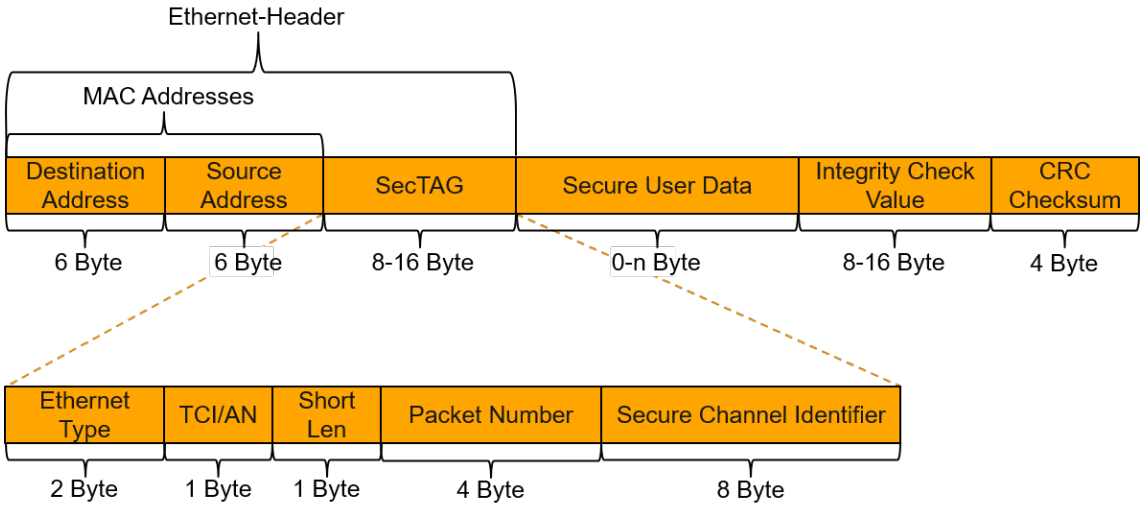

Abbildung 4: Ethernet Frame mit MACsec

Diese Auflistung beschreibt kurz die Bedeutung der einzelnen Blöcke aus dem Segment *SecTAG*:

- Ethernet Type: Gibt den Typen des Kommunikationsprotokolls an [5]
- TCI/AN:
  - TCI: Tag Control Information. In diesem Block ist unter anderem die Versionsnummer des MACsec-Protokoll enthalten.
  - AN: Association Number identifiziert bis zu vier sichere Verbindungen über einen gesicherten Kanal.
- Short Length: Ganzzahliger Wert, der die Anzahl der Oktette zwischen dem letztem Oktett von SecTAG und dem ersten Oktett von der Integrity Check Value angibt.
- Paket Number: Eindeutige Nummer des gesendeten Pakets.
- Secure Channel Identifier: Anhand dieser ID kann der sichere Kanal identifiziert werden.

2.3 Netzwerkmanagement im Automotiv-Bereich

Dieser Abschnitt erklärt den aktuellen und zukünftigen Stand der Technik in einem moderneren Fahrzeug.

Da die Größe der zu sendenden Daten in einem Fahrzeug ständig weiter steigt, ist es notwendig eine neue Generation von fahrzeug-interner Netzwerkinfrastruktur zu entwickeln. Die bisher genutzten Techniken, wie CAN, LIN und FLexRay sind für diese Datenmengen nicht mehr ausreichend. Aus diesem Grund existiert die Bestrebung Ethernet in das Fahrzeug zu integrieren, welches bereits unabhängig von der Automobilbranche unter der IEEE 802.3 [15] standardisiert wurde. Ethernet ist ein vielversprechender Ansatz, da es sich durch eine hohe Wiederverwendbarkeit für Komponenten, Software und Tools auszeichnet. Ebenso verfügt es über eine ausreichend dimensionierte Bandbreite und gute Skalierbarkeit des gesamten Netzwerkes. Der Hauptvorteil von Ethernet ist, dass jeder Teilnehmer seine eigene Leitung besitzt und anders als bei den Bus-Systemen kein geteiltes (eng. *shared*) System ist. Hierbei nutzen alle Teilnehmer dieselbe Leitung. Im Automobilbereich spielen aber auch noch andere Kriterien eine Rolle, wie beispielsweise Kosten, Robustheit und der Stromverbrauch [24]. Anhand dieser Kriterien muss sich Ethernet erst noch gegen seine bereits im Fahrzeug fest etablierten Konkurrenten durchsetzen. Diese Systeme erfüllen die Anforderungen an Kommunikation in Echtzeit zwischen Geräten. Da Ethernet

jedoch anders als Bus-Systeme ein verbindungslos Netzwerkmedium ist, muss auf eine andere Weise sichergestellt werden, dass Daten in Echtzeit ankommen. Dies wird mittels Time-Sensitive Networking (TSN) erledigt, welches im Unterabschnitt 2.4 näher erläutert wird.

In der nachfolgenden Tabelle 2.1 werden aktuelle und geplante Standardisierungen im Bereich Netzwerktechnologien aufgezeigt. Es werden Eckdaten zu Bandbreite, Erscheinungsjahr und Normen angegeben, unter denen weiter Informationen erlangt werden können. Unter Zuhilfenahme der einzelnen Standardisierungen, wird versucht ein möglichst effizientes Netzwerk zu erstellen.

Standard	Bandbreite	Name	Veröffentlichung
ISO-Norm 17987-1	25kbit/s	LIN BUS	1998
ISO 11 898	1Mbit/s	CAN BUS	1986
IEEE 802.3i	10Mbit/s	10Base-T	1990
ISO 17458-1/5	10Mbit/s	FlexRay	2000
IEEE 802.3u	100Mbit/s	100Base-TX	1995
IEEE 802.3ab	1000Base-T	1Gbit/s	1999
IEEE 802.3an	10Gbit/s	10GBase-T	2002
IEEE 802.3ba	100Gbit/s	100GBase	2010
IEEE 802.3bs	200Gbit/s	200GBase	2017
	400Gbit/s	400GBase	2017

Tabelle 2.1: Aktuelle Standards in modernen Fahrzeugen

2.4 Time-Sensitive Networking (TSN)

Im Jahr 2012 wurde die bisherige Standardisierung Audio Video Bridging (AVB) in einer neuen Gruppe unter dem Namen Time-Sensitive Networking (TSN) weiter entwickelt. Diese beschäftigen sich weiterhin mit der Standardisierung von Erweiterungen zur Erhöhung der Güte des Gesamtsystems.

Der Grund für den Wechsel von AVB auf TSN war, dass nicht nur Audio- und Video-Daten versendet werden sollten, sondern auch Kommunikationsdaten, wie z.B. Datenpakete für autonomes Fahren, wie Radardaten oder Sensordaten. Das Ziel ist es, Echtzeitfähigkeit im Bereich Ethernet-Netzwerke zu ermöglichen. Dies soll mit einer möglichst geringen Latenzzeit von maximal 100μs auf 5 Hops bei einer Datenrate von 100MBit/s zwischen den Netzwerkgeräten ermöglicht werden. Um das Versenden von Daten mit einer Latenz-Garantie zu ermöglichen, müssen wichtige Pakete mit einer höheren Priorisierung Daten gesendet werden als andere. In der folgenden Tabelle 2.2 werden die bisher wichtigsten Standards der Institute of Electrical and Electronics Engineers (IEEE) zum Thema TSN aufgelistet.

IEEE-Standard	Titel
802.1AS-Rev	Timing and Synchronization for Time-Sensitive Applications [7]
802.1Qbu	Frame Preemption [10]
802.1Qbv	Enhancements for Scheduled Traffic [**?**]
802.1Qca	Path Control and Reservation [11]
802.1CB	Frame Replication and Elimination for Reliability [8]
802.1Qcc	Stream Reservation Protocol (SRP) Enhancements and Performance Improvements [12]
802.1CM	Time-Sensitive Networking for Fronthaul [9]
802.1Qci	Per-Stream Filtering and Policing [13]
802.1Qca	Path Control and Reservation [11]
802.1Qcr	Bridges and Bridged Networks Amendment: Asynchronous Traffic Shaping [14]

Tabelle 2.2: Aktuell IEEE Standards zu TSN

In den folgenden Unterabschnitten werden die wichtigsten Themen von TSN detaillierter behandelt, die für ein echtzeitfähiges Netzwerk von besonderer Bedeutung sind. Es wird nicht auf alle Standards komplett eingegangen, da dies den Rahmen der Abhandlung sprengen würde. Bei Bedarf kann dies unter der IEEE 802.1 nachgelesen werden. Hier soll ein grundsätzliches Verständnis für die wichtigsten Techniken gegeben werden, um verschiedene Entscheidungen in der Arbeit nachvollziehen zu können.

2.4.1 Scheduling und Traffic Shaping

Alle teilnehmenden Netzwerkgeräte arbeiten bei der Bearbeitung und Weiterleitung von Datenpaketen nach den gleichen Regeln. TSN nutzt dafür unter anderem den IEEE Standard 802.1Qcc. Beim Shaping bezieht man sich hauptsächlich auf zwei Arten von Shapern. Diese werden in den nächsten beiden Unterpunkten kurz erläutert.

- **Time Aware Shaper (TAS)**
  Der IEEE802.1Qbv Standard definiert die Technik des TAS. Dieser stellt einen zeitlich gesteuerten Weiterleitungsmechanismus dar und ermöglicht die genaue Planung des Datenverkehrs im Netzwerk. Durch den Einsatz des Shapers können Verzögerungen kritischer Pakete durch Querverkehr vermieden werden. Er kann Zeiten bis zu 100µs garantieren. Das Prinzip von TAS basiert auf das Zeitscheibenmultiplexing-Verfahren, wodurch der Zugriff auf ein Medium zeitgesteuert abläuft. Der Shaper unterteilt die vorhandenen Zeiten in Intervalle. Hier gibt es sog. kritische- und nicht kritische-Zeitintervalle. Diese sind in verschiedene Warteschlangen unterteilt.
  Laut Definition im Standard kann ein Port bis zu acht Warteschlangen besitzen. Jede Queue bekommt ein sog. *Gate* (dt. Zeitfenster). Dieses nutzt der TAS um nur bestimmte Schlangen in einem Zeitintervall das Versenden von Daten zu erlauben. Die Übertragung kommt nur dann zustande, wenn das Gate geöffnet ist und noch genügend Zeit zur vollständigen Übermittlung des Frames bleibt.

- **Credit Based Shaper (CBS)**
  Der Credit Based Shaper ist ebenfalls in IEEE802.1 standardisiert. Die eingesetzte Technik garantiert eine Latenzzeit von 2 Millisekunden auf sieben Hops bei einer Datenrate von 100 Mbit/s. Er kann mit einem oder mehreren Warteschlangen eines Ports arbeiten. Der Mechanismus unterbricht niedriger priorisierte Datenpakete für wichtigere Frames. Jede Warteschlange bekommt *Credits*(dt. Guthaben). Ist kein Frame in der Queue, hat diese den Wert 0. Das Guthaben erhöht sich mit jedem Paket in der Schlange. Ist ein Frame aus einer Queue komplett versendet, verringern sich die Credits wieder.

2.4.2 Frame Preemption

Der Vollständigkeit halber wird hier noch Frame Preemption kurz erläutert. Es ist ebenfalls ein IEEE Standard und kann Verzögerungen bei der Latenzzeit von Paketen reduzieren. Zum aktuellen Zeitpunkt wird es aber noch nicht von den Endgeräten im Fahrzeug unterstützt und ist in diesem Zusammenhang nur ein weitere Möglichkeit die Latenzzeit zu senken.

Um Frame Preemption zu ermöglichen wird die Übertragung von weniger kritischen Paketen für kritische Pakete unterbrochen. In Abbildung 5 wird das Senden von Paketen mit und ohne Preemtion veranschaulicht. Frame 1 wird mit einer niedrigen Priorität wird bei t0 abgeschickt und der Frame 2 hat eine hohe Priorität und wird zum Zeitpunkt t1 gesendet. Wenn Preemtion aktiv ist, wird das Senden von Frame 1 unterbrochen und später fortgesetzt. Ohne diese Technologie muss der Frame 2 warten, bis der Frame 1 komplett gesendet ist.

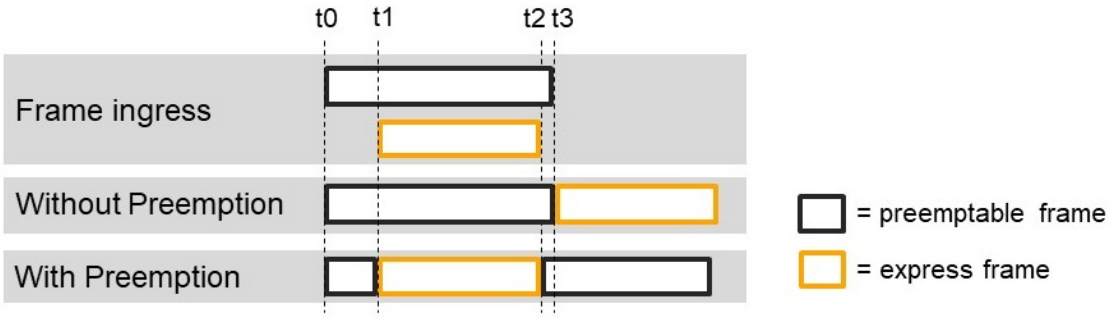

Abbildung 5: Frame Preemtion

Das ganze Prinzip muss auch hardwareseitig von Sender und Empfänger unterstützt werden. Frame Preemtion kann auf verschiedenen Ebenen der ISO/OSI-Modell stattfinden. In den unteren Schichten, wie beispielsweise auf der physikalischen Schicht 1, ist die IEEE 802.3 zuständig. Diese hat den den Standard unter der 802.3br verankert. Für alle höheren Schichten ist die Arbeitsgruppe IEEE 802.1 zuständig [6] [15].

Frame Preemtion wird vor allem in Kombination mit dem Time Aware Shaper verwendet. Hiermit können sog. Überlast-Situationen vermieden werden, bei denen es zum Überlauf der Warteschlangen im Switch kommen kann. Bei diesem Verfahren wird die sichere Versendung von Daten mit der gegeben Latenz-Anforderung garantiert. Pakete mit einer großen Latenzzeit können innerhalb der nicht kritischen Zeitintervalle des TAS versendet werden. Ein übermäßiges Senden von höher priorisierten Daten kann trotzdem zum Datenstau bei den Niedrigeren führen.

2.5 Sicherheit einer Anwendung

Der Punkt Savety ist ein großer Bereich in der Automobilbranche. Diese Arbeit betrachtet die grundsätzlich einzuhaltenden Parameter einer Anwendung nach ISO26262 [19]. Dies ist eine Norm für sicherheitsrelevante elektronische Systeme im Fahrzeugbereich. Dort werden verschiedene Vorgehensmodelle angegeben, welche die anzuwendende Methoden in der Entwicklung von Produkten im Kraftfahrzeugbereich definieren. Die Umsetzung der Norm soll eine funktionale Sicherheit im System gewährleisten. Da das Thema Sicherheit in Automobilsystemen jedoch ein großes Einzelthema ist, wird dies nur am Rande in der Arbeit behandelt. Eine Ausführliche Erläuterung des Themas kann im Buch *Functional Safety for Road Vehicles* nachgelesen werden. TODO CITE

2.6 Schutzziel einer Anwendung

In der Informationssicherheit gibt es das Konzept der Schutzziele. Dieses stellt sicher, dass sicherheitskritische Daten geschützt und unverändert übertragen werden. Der Zugriff auf die Daten ist zu beschränken und zu kontrollieren, sodass nur autorisierten Nutzern ein Zugriff gewährt wird. Die Schutzziele, die diese Anforderungen präzisieren, sind Integrität, Verfügbarkeit und Vertraulichkeit. Netzwerkgeräte, die auf kritische Daten zugreifen sollen, müssen je nach Sicherheitsanforderung eindeutig identifiziert werden können. Die entsprechende Eigenschaft der Geräte nennt man Authentizität, welches ein kombiniertes Schutzziel aus mehreren Parametern ist. Es ist eines der wichtigsten Ziele, da über die Authentifizierung der Zugriff auf Daten geregelt werden kann und somit die Verfügbarkeit gewährleistet wird [22].

Die angesprochenen Schutzziele werden in den folgenden Unterkapiteln kurz erläutert.

2.6.1 Integrität

Ein System gewährleistet die Datenintegrität, wenn es keinem Netzwerkteilnehmer möglich ist, die zu schützenden Daten unautorisiert zu verändern oder zu manipulieren. Dies kann durch Festlegen von Rechten an Daten erfolgen. So kann eine Anwendung beispielsweise nur Leserechte an den übermittelten Daten haben, während eine andere sie auch bearbeiten darf. Manipulation muss jederzeit nachgewiesen werden können und darf nicht unbemerkt bleiben. Dies kann z.B. anhand der bereits im Abschnitt 2.2.1 und 2.2.2 erwähnten Check-Summen überprüft werden. Sind die Summen nicht korrekt, weist es auf eine Veränderung der Daten hin. Eine Veränderung kann sich einerseits durch einen Übertragungsfehler ergeben oder auch durch Manipulation der Daten.

2.6.2 Verfügbarkeit

Das Netzwerk gewährleistet eine gute Verfügbarkeit, wenn authentifizierte und autorisierte Netzwerkteilnehmer bei der Ausführung ihrer Aktionen nicht unautorisiert beeinträchtigt werden können. Die Erfüllung dieses Schutzziels benötigt jedoch eine Nutzungsverwaltung von Systemressourcen und ist nicht Teil dieser Arbeit.

2.6.3 Vertraulichkeit

Ein System weist eine Informationsvertraulichkeit auf, wenn es keine unautorisierten Informationsgewinnung ermöglicht. Dies wird mittels Kontrolle der Datenflüsse im Netzwerk gewährleistet. Eine Möglichkeit hierzu ist die Verwendung von einer Ende-zu-Ende Verschlüsselung. Auch dies erfordert eine eigene Technologie, welche nicht Teil dieser Arbeit ist.

2.6.4 Authentizität

Die Authentizität eines Objekt gibt dessen Echtheit und Glaubwürdigkeit an. Anhand dieser kann sicher gestellt werden, dass es sich um einen berechtigten Nutzer der Daten handelt. Dies wird durch eindeutige Identifikationsmerkmale, wie beispielsweise ID, gewährleistet.

Kapitel

3

Anforderungen an die Konfiguration

Das folgende Kapitel behandelt die Anforderungen an ein Netzwerk und die Parameterdefinition für das applikationsspezifische Manifest. Dieses soll möglichst wenig Elemente beinhalten, von denen weitere Parameter automatisch abgeleitet werden können. Die Ableitung soll an einer zentralen Stelle im System erfolgen und nicht mehr manuell beeinflusst werden müssen. Die wiederum abgeleiteten Parameter sollen ein Netzwerk eindeutig spezifizieren und somit eine Konfiguration möglich machen. Diese erfolgt durch einen Managers, auf den im Laufe der Arbeit noch genauer eingegangen wird.

Alle bereits in den Grundlagen angesprochenen Konzepte werden als Grundlagen für die Parametererstellung vorausgesetzt.

3.1 Netzwerkübersicht

In der Zieldefinition der Arbeit wurde bereits ein grober Überblick über die Funktion des Netzwerk-Managers im System gegeben, welcher jedoch nicht die einzige Komponente im Netzwerk ist. Deshalb wird nun die Sicht auf das Netz nur noch etwas näher spezifizieren. Der Manager wird von anderen Netzwerkkomponenten mit Eingaben befüllt, anhand derer er Entscheidungen treffen kann. In Abb. 6 wird ein einfaches Netzwerk mit wenigen Komponenten dargestellt. Eine Anwendung soll im System gestartet werden. Dies benötigt eine Kommunikation zweier ECU's die mittels einem Switch verbunden sind. Um die Kommunikationsstrecke nach den geforderten Bedingungen der Applikation zu konfigurieren, müssen der Applikations-Orchestrator, welcher für die Verwaltung der Anwendungen verantwortlich ist und der Netzwerk-Manager gemeinsam eine Konfiguration für das Netzwerk generieren. Beide Komponenten werden detaillierter im Kapitel 4 vorgestellt, in dem noch spezifischer auf das Netzwerk eingegangen wird. Vorerst soll der generelle Überblick über das System genügen.

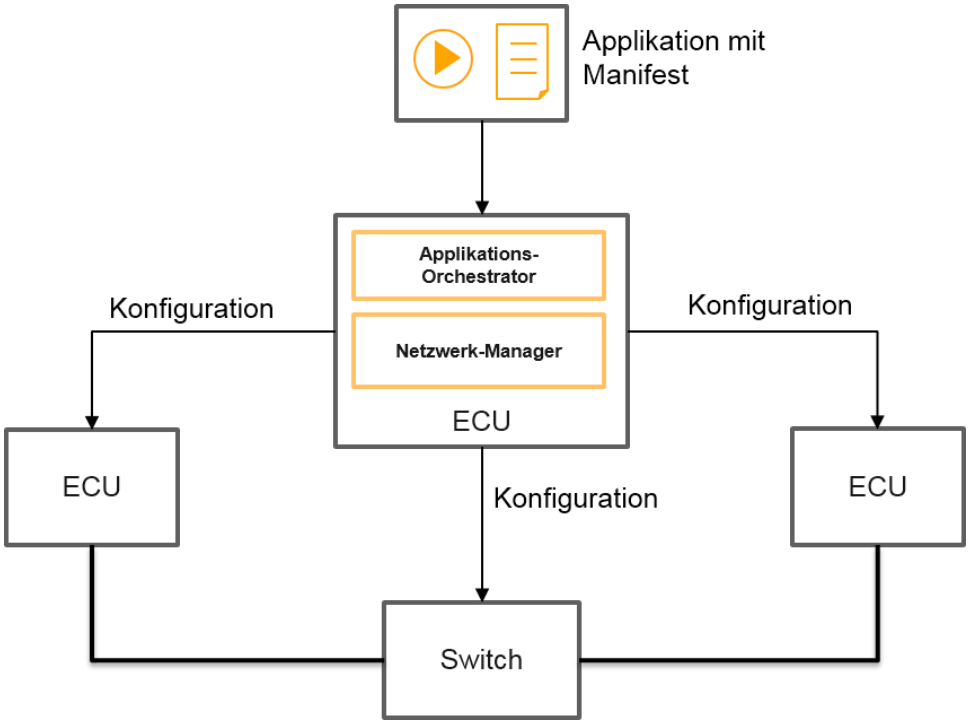

Abbildung 6: Darstellung eines vereinfachten Netzwerkes

3.2 Anforderungen an das neue Netzwerk

Generell sollte ein Netzwerk alle gängigen Sicherheitsanforderungen erfüllen. In der Automobilbranche ist dies besonders wichtig, weshalb unter der Norm ISO26262 die wichtigsten Störungsmöglichkeiten definiert sind. Sie müssen erfüllt sein, um ein Netzwerk nach den sog. Safety-Anforderungen zu erstellen [17]. In der Tabelle 3.1 sind die wichtigsten Störungsfälle im System aufgelistet. Anhand dieser Anforderungen, welche als Grundvoraussetzung für sichere System gelten, werden im Folgekapitel die Parameter für das Applikationsmanifest festgelegt. Wenn bereits einige dieser Anforderungen nicht abgedeckt werden, wird es schwierig ein sicheres Netzwerk aufzubauen.

Störung	Erläuterung
Korrupte Nachricht	Die empfangenen Daten einer Nachricht sind falsch
Nachrichten Delay	Nachricht wird später als erwartet empfangen
Nachrichtenverlust	Nachricht oder Paket geht im System verloren
Wiederholte Nachricht	Empfänger erhalten zwei oder mehr Nachrichten mit dem selben Inhalt
Nachfolgeregelung	Nachrichten werden in falscher Reihenfolge empfangen
Unterbrechungen	Empfänger erhalten Nachrichten, die sie nicht erwarten
Maskierung	Nachricht wird unter Verwendung einer falschen ID versandt
Asymmetrische Auskünfte	Nachrichten von einem einzigen Absender werden empfangen (kann auch an eine Gruppe an Empfängern gehen)
Abgesicherte Kommunikation	Verhindert den unbefugten Zugriff auf Kommunikationsdaten

Tabelle 3.1: Übersicht der Manifest-Parameter

3.3 Manifest-Parameter

Eine Applikation hat verschiedene Anforderungen an ihre Netzwerkumgebung, die durch Parameter definiert werden. Im ersten Schritt werden gängige Netzwerk-Parameter anhand der in den Kapitel 2 definierten Werte und Techniken, insbesondere im Hinblick auf die Anforderungen an Echtzeitsystem und der Schutzziele eines Systems aus Abschnitt 2.6 ausgewählt. Dies mussten im nächsten Schritt mit den Anforderungen aus dem vorherigen Punkt 3.2 vereinigt werden. Nur wenn alle Punkte erfüllt werden, kann ein Netzwerk überhaupt im Hinblick auf Sicherheit weiter behandelt werden. Welcher Parameter für die Vermeidung eine der erwähnten Störungen verantwortlich ist, wird in der Tabelle 3.2 mit dargestellt. Es wird jeder Parameter mindestens einem der Störungsfälle zugeordnet, um die direkte Abhängigkeit untereinander darzustellen. Viele der aufgelistet Parameter sind jedoch nicht nur für ein Ziel zuständig, sonder werden über mehrere hinweg benötigt.

Der letzte Schritt in der Parameterdefinition ist das weitest mögliche abstrahieren der netzwerkspezifischen Eigenschaften. Hierdurch konnten einige wenige Parameter definiert werden, von denen wiederum der Netzwerkmanager (beschrieben in Kapitel 4.1) alle für eine Konfiguration wichtigen Parameter ableiten kann. Dies ermöglicht eine Entwicklung einer Applikation, unabhängig von einer konkreten Netzwerktechnologie. Somit können auch spätere Änderungen in der Netzwerkinfrastruktur vorgenommen werden, ohne die Anwendungen überarbeiten zu müssen. Auf diesen Weg erreicht man den Schritt von einem normalen homogenen Netzwerk zu einem heterogenen Netzwerk. Der Unterschied ist in erster Linie, dass anders als bei einem homogenen Netz, auch unterschiedliche Softwaresysteme untereinander kommunizieren können [4].

Jeder Entwickler legt für seine Anwendung die gewünschten Parameter fest, welche sich im Manifest befinden. Um einen fehlerfreien Ablauf zu garantieren sind diese, wie schon erwähnt, lediglich abstrahierte Parameter die später softwareseitig in die benötigten Parameter aufgeschlüsselt werden. Das entstandene File wird als Applikationsmanifest bezeichnet und ist im Format JavaScript Object Notation (JSON). JSON ist ein Datenformat, welches zum Datenaustausch zwischen Anwendungen verwendet wird. Es ist für den Entwickler einfach les- und schreib-bar, was eine einfache Bedienung verspricht. Die strukturelle Form von dieser Sprache lässt in einfacher Form komplexe Strukturen abbilden. Dies ermöglicht es beispielsweise ein Array oder Liste an geschachtelten Elementen zu übermitteln [3]. Dies wird ist von Vorteil, falls eine Applikation mehre verschieden Arten von Nachrichten übermitteln muss. Wäre dies nicht möglich könnte eine Anwendung immer nur genau eine Kommunikationspater haben und eine Art von Daten versenden.

In der Tabelle 3.2 enthält die Parameter, welche aus Netzwerksicht gesetzt werden müssen, um eine Konfiguration der gesamten Kommunikationsstrecke zu garnieren. Die werden in den folgenden Unterkapiteln näher erläutert.

Parameter	Spezifikation	Störung
name	Name der Applikation im System	Maskierung
id	Eindeutige Identifikationsnummer	Maskierung
groupId	Eindeutige Gruppenidentifikationsnummer	Asymmetrische Auskünfte
domain	Domain einer Anwendung	Asymmetrische Auskünfte
cyclic	Zyklisch- oder Event-gesteuerte Anwednung	Wiederholte Nachricht Nachfolgeregelung Unterbrechungen
timesync	Zeitsynchronisation erforderlich	
maxLatency	Latenzzeit einer Nachricht	Nachrichten Delay
minLatency	Unter Grenze der Latenzzeit (Jitter)	Nachrichten Delay
messageSize	Größe der zu sendende Daten	Nachrichtenverlust
frequency	Wiederholungsgeschwindigkeit einer Nachricht	Nachrichtenverlust
integrity	Korrektheit der zu sendenden Daten	Korrupte Nachricht Abgesicherte Kommunikation

Tabelle 3.2: Übersicht der Manifest-Parameter

3.3.1 Identifikation und Gruppen

Jede Anwendung besitzt eine ID anhand dieser sie eindeutig identifiziert werden kann. In dem Konzept der Erweiterbarkeit des Netzwerkes muss dies ID weltweit eindeutig hinterlegt werden, um eine Anwendung unmissverständlich einer Herkunft zuordnen zu können. Vor allem wenn es darum geht Fehler im System zu finden ist eine eindeutige Zuordnung wichtig, da sich ein *Bugfix* der Anwendung als schwierig herausstellen kann. Dieser muss nicht nur lokal im System erfolgen sonder weltweit für alles Fälle in denen dies Applikation zum Einsatz kommt.

Neben der Eliminierung der Fehlermöglichkeit Maskierung, ist ein positiver Nebeneffekt einer eindeutigen ID, die Regelung der Kommunikation zwischen den Applikationen innerhalb eines Netzwerkes. Das Format der ID ist definiert als *id-12345* und muss einmalig im System sein. Der Netzwerkmanager besitzt ein Verzeichnis über alle laufenden Anwendungen, anhand dieser können Kommunikationsstrecken aufgebaut werden.

Da es auch Applikation gibt, die gemeinsam benachrichtigt werden müssen, existiert eine Gruppenidentifikationsvariable. Angelehnt an das Format der ID hat die Gruppen-ID die Form *group-12345*. Anhand dieser können Multicast-Nachrichten an alle Anwendungen einer Gruppe gesendet werden. Ebenso kann eine Domain angeben werden, um eine Applikation eindeutig zuordnen zu können.

3.3.2 Sender und Empfänger

Durch die eindeutige ID einer Anwendung, können Regeln für ein- und ausgehende Nachrichten definiert werden. Da eine Anwendung Unicast- und Multicast-Nachrichten an einen Empfänger senden kann, oder auch von ihm empfängt, müssen nachrichtenspezifische Parameter vergeben werden. So kann die maximale Latenzzeit bei Nachricht *N1* vom Sender *id-11111* zu Empfänger *id-11112* anders sein, als die der Nachricht *N2* zu Empfänger *id-11113*. Die maximale Latenz eines Datenpakets definiert auch, wie lange dieses gültig ist. Die Gültigkeit eines Pakets richtet sich nach der Zeit zwischen erstem und erneutem Senden. So hat Nachricht *N1* eine Gültigkeitsdauer von 200ms. Da Sender und Empfänger verschiedene Anforderungen an die Latenz haben können, müssen immer beide Seiten betrachtet werden und die geringere Zeit wird für die Kommunikation gesetzt.

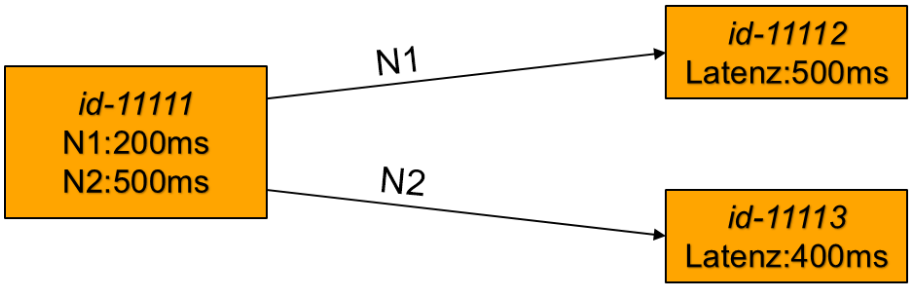

Abbildung 7: Nachrichtenspezifische Parameter

Da *id-11111* die Nachricht an *id-11112* mit geringer Latenz sendet als benötigt, wird dies für *N1* als Wert gesetzt. Bei der Nachricht an *id-11113* wird eine niedrigere Latenz vom Empfänger erwartet, als *id-11111* sendet. Somit muss die Latenzzeit der Nachricht *N2* auf 400 ms angepasst werden:

3.3.3 Art der Nachrichten

Es gibt im System zwei verschiedene Arten von Nachrichtenaustausch.

- **Zyklische Nachrichtenaustausch**
  Nachrichten die in fest definierten Zeitabständen von einer Applikation gesendet werden, bezeichnet man als zyklisch. Die Abstände werden mit der Frequenz einer Nachricht angegeben. Wenn eine zyklische Laufzeit angegeben wird, muss zusätzlich im System eine laufende Paketnummer für gesendete Daten vergeben werden, um Fehler wie wiederholte Nachrichten, vertauschte Reihenfolge der Pakete oder Unterbrechung durch beispielsweise Frame Preemtion zu vermeiden.

- **Event-gesteuerte Nachrichtenaustausch**
  Sende eine Applikation nur bei bestimmten Ereignissen eine Nachricht, wird dies event-gesteuerte bezeichnet. Sobald diese jedoch gestartet sind, könne sie ebenso wie bis zu einem definierten Ende wiederholt ausgeführt werden.

3.3.4 Datenrate

Eine Applikation benötigt einen bestimmten Anteil von der Bandbreite einer Leitung im Netzwerk. Die benötigte Datenrate kann mittels der beiden folgenden Parameter errechnet werden.

- **Nachrichtengröße**
  Die Nachrichtengröße definiert die Größe einer zu sendenden Nachricht an einen anderen Teilnehmer im System. Diese wird in *byte* angeben.

- **Frequenz einer Nachricht**
  Jede Nachricht wird in bestimmten Abständen nacheinander versendet. Dies wird in Nachrichten pro Sekunde angegeben. Wie bereits einen Absatz weiter oben beschrieben, gibt es jedoch auch Nachrichten ohne Frequenz, da dies nur einmalig gesendet werden. Hierbei ist die Frequenz mit dem Wert eins zu belegen.

Die Bandbreite pro Nachricht errechnet sich somit anhand der Formel:

$$Bandbreite = Nachrichtengroesse * Frequenz \tag{3.1}$$

Eine Applikation kann mehrere ausgehende Nachrichten versenden. Diese werden gesondert von einander betrachtet und im Nachhinein aufsummiert. Im Manifest werden die verschieden Nachrichten einzeln angegeben, jede Nachricht besitzt ihre eigenen Parameter. Falls eine Anwendung auch eventgesteuerte Einzelnachrichten sendet, empfiehlt es sich

einen Anteil der Bandbreite der Leitung frei zu halten. Wird dies nicht gemacht, kann es zu Datenstau und den daraus resultierenden Paketverlust kommen. Dies gilt es zu berücksichtigen und zu vermeiden. So wird eine relativ genaue Abschätzung der benötigten Bandbreite ermöglicht.
Wenn mehrere Anwendungen auf einer ECU laufen, kann anhand ihrer Einzelbandbreiten die gesamte Auslastung einer Leitung berechnet werden. Anhand der Gesamtauslastung kann die benötigte Bandbreite einer Applikation zugesichert werden, damit diese auf einem Steuergerät laufen kann.

3.3.5 Latenz und Jitter

Die Latenzzeit (Kapitel 2.1.3) wird angegeben mit dem Parameter *maxLatency* und ist einer der wichtigsten Parameter für die Netzwerkkonfiguration. Dieser gibt, wie bereits beschrieb, die maximale Dauer bis zur Ankunft einer Nachricht im System an. Um die gewünschte Zeit zu erreichen, müssen Technologien wie Scheduler und Tarffic Shaper (Kapitel 2.4.1) eingesetzt werden. Diese können anhand der abgeleiteten Parameter aus der Latenz aufgesetzt werden. Die Latenz einer Nachricht wird im System wie folgt berechnet:

$$Latenz = Ankunftzeit - Startzeit \tag{3.2}$$

Kann ein System die geforderte zeitliche Anforderung einer Anwendung nicht einhalten, kann diese nicht in dem Netzwerk betrieben werden. Aus diesem Grund sollte bereits bei der Entwicklung einer Applikation die gegebene Infrastruktur des Endsystems betrachtet werden.

Wie bereits in Kapitel 2.1.4 erwähnt, wird die Schwankung (eng. Jitter) als die maximale Abweichung der Ankunftszeit einer Nachricht bezeichnet. Diese kann sowohl positiv als auch negativ sein, da dieser ein Zeitfenster mit ober und unter Schranke angibt. Der Differenz zwischen ober und unter Grenze ergibt den Jitter. In Manifest wird die Latenz (*maxLatency*) als ober Schranke benutzt. Die unter ist im Parameter *minLatency* definiert. Die Abweichungszeit berechnet sich mit der Formel:

$$Jitter = |maximaleLatenz - minimaleLatenz| \tag{3.3}$$

Das errechnete Zeitfensters gibt an, in welchem zeitlichen Bereich die Nachricht beim Empfänger ankommen muss. Ist keine minimale Latenz angegeben, reicht es lediglich die maximale Latenz als Obergrenze der Ankunftszeit zu setzen. Dies bedeutet, dass eine Nachricht lediglich in eine bestimmten Zeit ankommen muss. Es ist jedoch egal ob sie früher ankommt. Durch Jitter kann ein zu frühes ankommen der Daten verhindert werden, da hier eine Untergrenze existiert.

3.3.6 Zeitsynchronisation

Die Eigenschaft Zeitsynchronisation (eng. time synchronisation) ist notwendig um einen gemeinsamen Zeitstempel über verschiedene Anwendungen zu garantieren. Die Synchronisation der Applikationen kann über unterschiedliche Mechanismen erfolgen. In dieser Arbeit wird jedoch nur auf das Precision Time Protocol (PTP) eingegangen. Dieses wurde von der IEEE unter der Nummer 1588 standardisiert. In diesem Standard sind fünf Arten von Uhren definiert, anhand dieser eine Zeitsynchronisation stattfinden kann. Im Automotiv-Bereich werden lediglich zwei der fünf verwendet.

- **Ordianary Clock**
  Diese Uhr wird als die „einfache Uhr“ bezeichnet und kann sowohl als Master Clock, als auch Salve Clock im Netzwerk eingesetzt werden. Sie besitzt eine physische Schnittstelle, die zum Senden und Empfangen von Ereignissen verwendet wird. Eine Ordianary Clock im „Mastermode“ kann ihre Zeit frei definieren, oder auch anhand eines Zeitservers synchronisieren (z.B. GPS Zeiten). Im „Slavemode“ folgt sie den Zeitdaten eines anderen Masters im System

- **Boundary Clock**
  Diese Uhr wird in der Regel als Slave einer Master Clock verwendet. Sie ist oft in Netzwerkkomponenten integriert, z.B. in Switchen oder Repeatern. Dort kann die Uhr Zeitdaten als Slave empfangen und als Masterzeit weiterleiten. Sie besitzt, anders als die Ordianary Clock , mehrere physische Schnittstellen. Jeder dieser Ports ist wie eine „einfache Uhr“ aufgebaut und kann auch so verwendet werden.

Die Genauigkeit einer Uhr wird anhand vordefinierter Hexadezimalwert spezifiziert. Diese werden bei der Konfiguration von einer Uhr mit angegeben um die gewünschte Exaktheit zu erreichen [18].

Es können auch mehrere Uhren und Genauigkeiten für eine Anwendung definiert werden. Dies ist nötig, wenn eine Applikation an verschieden Empfänger Daten übermittelt, die jedoch alle andere Anforderungen an Zeitgenauigkeit haben. Vor allem der zuvor erwähnte Time-aware Shaper (Kapitel 2.4.1) benötigt die exakte Zeit um auf seine Latenzzeit Garantie zu kommen.

Die Angabe erfolgt im Manifest mittels eines vierstelligen Strings:

- 1. Stelle: 0 = deaktiviert, 1 = aktiv
- 2. Stelle: 0 = Slave, 1 = Master
- 3. und 4. Stelle: Genauigkeit der Zeitdaten

Wert	Spezifikation	Wert	Spezifikation
20	Auf 25ns genau	29	Auf 1ms genau
21	Auf 100ns genau	2A	Auf 2,5ms genau
22	Auf 250ns genau	2B	Auf 10ms genau
23	Auf 1μs genau	2C	Auf 25ms genau
24	Auf 2,5μs genau	2D	Auf 100ms genau
25	Auf 10μs genau	2E	Auf 250ms genau
26	Auf 25μs genau	2F	Auf 1s genau
27	Auf 100μs genau	30	Auf 10s genau
28	Auf 250μs genau	31	Auf >10s genau

Tabelle 3.3: Genauigkeit der Zeitdaten im Netzwerk [18]

3.3.7 Sicherheit auf Richtigkeit der Daten

Ist die Richtigkeit der Daten wichtig gilt es die in Kapitel 2.6 zu gewährleisten. Dies kann aktuell nur anhand von Media Access Control Security umgesetzt werden. MACsec bietet die in Abschnitt 2.2.2 definierte Funktionalität welche beispielsweise eine Integrität der Daten sicher stellt. Auch eine Authentifizeirung der Kommunikationspartner ist bereits anhand der MAC Adressen möglich, da diese im System eindeutig sein müssen. Die weiteren genannten Schutzziel können mit dieser Technologie noch nicht erreicht werden. Dies müssen im Nachgang noch spezifisch betrachtet werden, um die Machbarkeit im jeweiligen System sicherzustellen. Eine Möglichkeit Vertraulichkeit zu gewährleisten wäre eine Verschlüsselung der Kommunikationsstrecke. Dies ist jedoch nur eine Möglichkeit und wird nicht weiter in dieser Arbeit verfolgt.

Der Parameter *integrity* ist lediglich eine boolische Variabel und gibt den Entwickler die Möglichkeit die Kommunikation zwischen den Endgräten zu Überwachen. Diese sog. Flag ist nicht generell gesetzt, da es Performance-Einbusen im System verursachten kann.

Kapitel

4

Automatisierte Konfiguration

Der Abschnitt automatisierte Konfiguration behandelt den bereits erwähnten Netzwerkmanager und dessen Funktionalität. Diese besteht in erster Linie darin das Netzwerk in einem Automobil für benötigte Anforderungen von Applikationen, zu konfigurieren. Eine zweite Komponente des Systems ist der Netzwerkorchestrator, welcher die Anforderungen einer Applikation an den Manager übergibt. Dieser wird ebenfalls in diesem Kapitel kurz erklärt.

Des weiteren werden die im Kapitel 3 definierten Parameter in die notwendigen Eigenschaften des Systems überführt. Jeder Eintrag im Manifest, kann wieder in Unterparameter aufgebrochen werden. Hierdurch wird die automatisierte Konfiguration des Fahrzeugnetzes ermöglicht.

4.1 Der Netzwerk-Manager

Eines der wichtigsten Komponenten im neuen Fahrzeugnetzwerke wir der Netzwerkmanager sein, dieser enthält alle nötigen Funktionen um das Netzwerkes zu konfigurieren. Er ist ein der zentralen Bausteine im System und kennt die gesamte Topologie des Fahrzeuges. Anhand dieses Kenntnisse und der vom Orchestrator übergeben Parameter, kann der Manager die sich im Fahrzeug befindenden Switche und ECU's konfigurieren. Die Funktionsweise des Netzwerkmanagers kann am besten mit dem Software-defined networking (SDN) Prinzip erläutert werden [25]. Die Grundidee hinter SDN ist die zentrale Verwaltung aller Netzwerkkomponenten in einem System. Hierbei kommt es zu einer Abstraktion der Schichten. Es wird die *Control Plane* (z. dt. Kontrollschicht) von der *Data Plane* (z. dt. Datenschicht) abgespalten. Die Kontrollschicht ist für die Steuerung der Datenschicht verantwortlich. Dies wird in der Abb. 8 vereinfacht dargestellt.

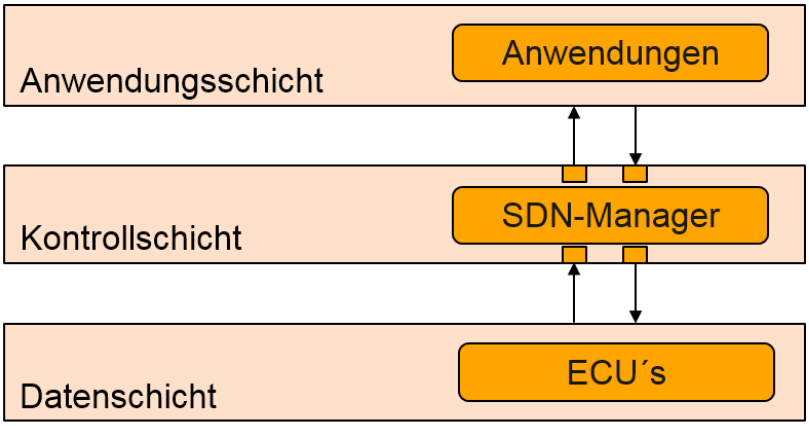

Abbildung 8: Aufbau der Schichten mittels SDN

In der mittleren Schicht befindet sich der Manager. Dieser sendet und empfängt mittels Schnittstellen Daten und Befehle. So kann beispielsweise eine Anwendung eine Netzwerkanforderung an den Manager senden. Dieser wiederum konfiguriert anhand der empfangenen Daten die entsprechenden ECU's. Das Prinzip wird auch im Fahrzeugnetz weiter verfolgt und angewendet.

In Abbildung 9 wird der Netzwerkmanager im Fahrzeug veranschaulicht. Er besteht aus vier Hauptkomponenten (z. eng. Core Functionality):

- Path finding: z. dt. Pfadfindung, diese ist verantwortlich für das Finden des bestmöglichen Pfad im Netzwerk.
- Network Discovery: z. dt. Netzwerkerkennung, diese kümmert sich um die Netzwerktopologie, um diese immer aktuell zu halten.
- Interface to other components: z. dt. Schnittstelle nach außen.
- Remote Configuration: z. dt. automatische Konfigurationsschnittstelle, diese konfiguriert die ECU's nach den Anforderungen des Manifests.

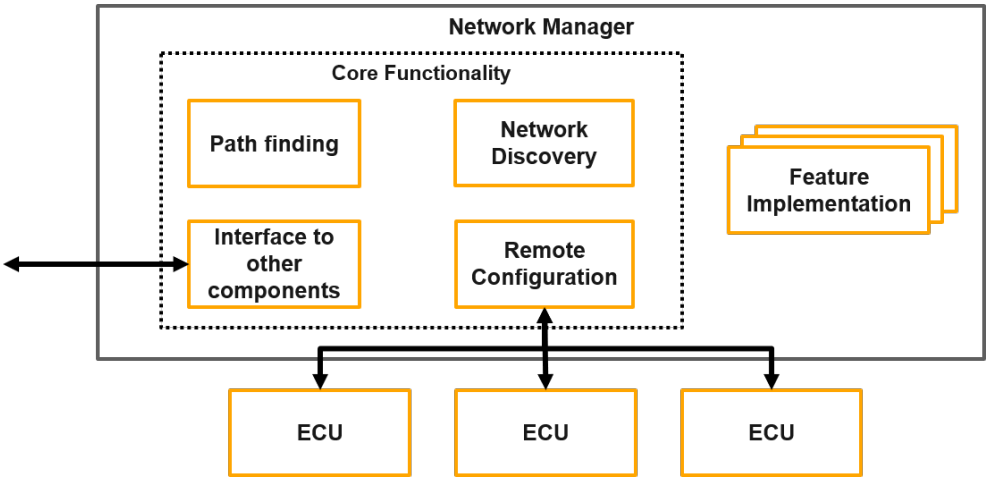

Abbildung 9: Aufbau des Netzwerkmanagers

Die ebenfalls im Bild dargestellten *Feature Implementations* stellen die einzelnen Funktionen des Managers da, mit denen er das Netzwerk konfigurieren kann. Werden Daten aus der Anwendungsschicht an die Schnittstelle des Managers übergeben, kann dieser mittels der implementierten Funktionen die entsprechende Konfiguration für das System erstellen. In den Funktionen sind alle nötigen Mechanismen hinterlegt, die eine ECU benötigt, um die Netzwerkanforderungen zu erfüllen. Ein Beispiel hierfür sind die in Kapitel 2.4 beschrieben Shaper, mit denen bestimmte Latenzzeiten eingehalten werden können.

Im weiteren Verlauf dieser Arbeit werden einzelnen Funktionen und die Konfigurationsschnittstelle implementiert und getestet. Die andern Komponenten des Netzwerkmanagers werden als gegeben betrachtet und sind nicht Teil der Ausarbeitung.

4.2 Der Applikations-Orchestrator

Der Orchestrator ist im System verantwortlich für die Verwaltung und Verteilung der Anwendungen auf den einzelnen ECU's. Er kennt nicht die gesamte Netzwerktopologie, sondern weiß lediglich welche ECU's und Anwendungen sich gerade im Netzwerk befinden. Der Applikations-Orchestrator ist in erster Linie für die Verwaltung der Funktionalität des Systems verantwortlich. Wenn eine neue Applikation gestartet werden soll, liest er die benötigten Parameter aus dem Applikationsmanifest aus und ermittelt eine Liste an geeigneten ECU's. Diese werden in absteigender Reihenfolge ihrer Tauglichkeit nach sortiert. In der Liste befinden sich alle Geräte, die für die Ausführung der Anwendung in geeignet sind. Die Auswahl wird anhand der benötigten Leistung einer Applikation getroffen, dies garantiert in erster Linie einen reibungslosen Ablauf der Funktion. Als Leistungsparameter werden beispielsweise Speicherbedarf und CPU Leistung angegeben.

Ist die Liste vollständig, übergibt er diese mit den Parameter aus dem Manifest an den Netzwerkmanager. Dieser prüft die gewünschten Endgeräte der Reihe nach auf ihre netzwerkseitige Tauglichkeit. Sobald eine der ECU's den Anforderungen entspricht, sendet er die ID des Gerätes an den Orchestrator zurück. Wenn dies der Fall ist, kann die Anwendung auf der zugehörigen ECU gestartet werden. Sollte keines der Endgeräte in der Liste für den Betrieb geeignet sein, gibt der Manager einen Fehler zurück. Dies bedingt einen Abbruch des Versuches die Anwendung zu starten, im Orchestrator.

Abbildung 10 veranschaulicht den Findungsprozess einer geeigneten ECU für eine Applikation im Netzwerk. Hierbei müssen die beiden Komponenten, Orchestrator und Netzwerkmanager, miteinander kommunizieren.

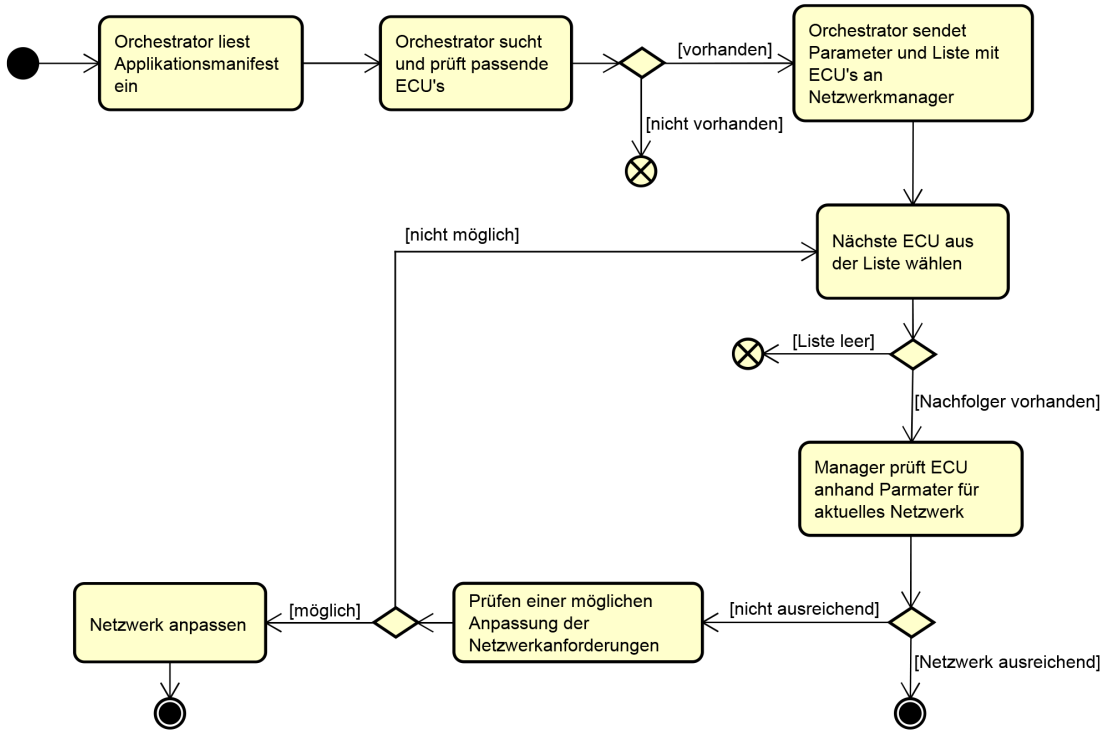

Abbildung 10: Kommunikation von Orchestrator und Manager

4.2.1 Verwaltung der Anwendungen

Der beschrieben Vorgang zur Hinzufügung einer Anwendung, wird jedoch nur eine der nötigen Funktionen des Orchestrators im Hinblick auf Applikationsverwaltung sein. Es soll nicht nur möglich sein, neu Anwendungen zu starten, sonder auch bereits laufende auf eine anderer ECU umzuziehen oder diese endgültig zu beenden. Dies kann nötig sein, falls eine Kommunikationsstrecke bereits stark ausgelastet ist, wobei andere noch genügend Kapazität hätten, um einen reibungslosen Ablauf des Systems zu garantieren. In diesem Fall würde eine Applikation umgezogen werden.

Dies ganzen Funktionalitäten werden an das Prinzip der Representational State Transfer (REST) Technologie angelehnt [21]. Dies zeichnet sich in erster Linie durch ihre vordefinierten Aktionsprinzip *CRUD* aus. Es steht für *create, retrieve, update und delete*. Drei der Funktionen können in den Prozess des Orchestrators überführt werden:

- *Create*, eine neue Anwendung wird im System gestartet.
- *Update*, der Status einer laufenden Anwendung im System soll geändert werden. Dies kann beispielsweise eine Umzug auf eine andere ECU sein. Auch das Pausieren einer Anwendung, ohne sie endgültig aus dem System zu entfernen, kann so realisiert werden.
- *Delete*, eine laufende Anwendung soll im System beendet und entfernt werden. Die Applikation wird dabei vollständig aus den System gelöscht.

Dies Arbeit kümmert sich lediglich um das Hinzufügen einer neuen Applikation in ein bestehendes System. Die andere Punkte müssen separat betrachte werden, da sie ein Wiederherstellen eines vorhergegangenen Netzwerkzustands bedingen. Hierzu gibt es verschiedene Ansätze, wie das Speichern der letzten Konfigurationen vor einer Erweiterung des Netzwerkes oder das komplette Neukonfigurieren des Systems. Die Ausarbeitung dieser Prinzipien würde jedoch den Rahmen der Abhandlung sprengen und wird hier nur der Vollständigkeit halber erwähnt.

4.3 Feature Entwicklung

Die im Abschnitt Netzwerkmanager (Kapitel 4.1) angesprochen Features, kümmern sich um die automatisierte Erkennung der Parameter und deren Weiterverarbeitung. In den folgenden Unterabschnitten wir jeder Eintrag aus dem Applikationsmanifest behandelt und definiert, welche Ableitungen von diesem gemacht werden können. Ein Hauptparameter kann in mehreren Features verarbeitet werden. Die Funktionen splitten nicht nur Parameter auf, sie definieren auch Mechanismen, die eine Einhaltung von applikations-spezifischen Anforderungen garantieren.

Der Aufbau der einzelnen Unterkapiteln wird nach einem vorgegebenen Schema erfolgen:

- Definition des Features
- Erläuterung durch Verwendungsmöglichkeiten oder Aktivitätsdiagramm

Es wird bei jedem Parameter nur auf die Ableitungen eingegangen, die auch im Automobil-Bereich umgesetzt werden. Wenn bei der Ableitung der Paramet eine bestimmte Technik gefordert wird, die eine ECU oder Switch gewährleisten muss, wird der Einfachheit halber auf das Abprüfen im Diagramm verzichtet. Dies wird vor jeder Konfiguration der Geräte sowieso überprüft und liefert eine Abbruchbedingung zurück, falls die Technologie nicht zur Verfügung stehen sollte.

4.3.1 Feature Kommunikationsstrecken Aufbau

Eine Grundvoraussetzung für die Verständigung zwischen Endgeräten ist ein Aufbau einer Kommunikationsstrecke unter den Teilnehmern. Dies bedeutet eine Konfiguration aller Ports die ein Versendete Nachricht passieren muss. Je nach Anforderung aus dem Applikationsmanifest und der übermittelten Informationen aus dem Orchestrator, muss eine geeigneter Weg im System gefunden werden. Die Parameter *id, groupId, domain* und der MAC-Adressen der einzelnen Teilnehmer, können verschieden Einstellungen an den ECU's und Switchen vorgenommen werden. Nicht alle haben die selbe Technologie was bedeutet, dass endgerätspezifisch entschieden werden muss, welche Konfigurationen möglich sind.

Mögliche Verwendungen wären beispielsweise:

- Portforwarding, was eine Weiterleitung an einen anderen Port eines Endgerätes bewirkt

- Ingress Filtering, ist eine Art Firewall und schützt vor Unerlaubter Kommunikation (Definition in Abschnitt 5.1)

Hauptsächlich dienen die Port-Konfigurationen zum reibungslosen Datenaustausch. Sie können aber auch zu einen gewissen Teil zum einhalten der in Abschnitt 2.6 definierten Schutzziele beitragen. Durch definierter Portregelungen wird ein unerlaubter Datenaustausch erschwert.

4.3.2 Feature Echtzeitfähigkeit

Um das Feature Echtzeitfähigkeit umzusetzen, werden mehre Parameter benötigt. In *maxLatency* ist die geforderte Latenzzeit angegeben, dies ist eine der wichtigsten Eigenschaften für eine Anwendung in einem Netzwerk. Aus dieser können verschieden Einstellungen gefolgert werden. Das zweite Attribut ist die *minLatency*. Dieser ergibt, wie in Abschnitt 3.3.5 beschrieben, zusammen mit der *maxLatency* den geforderten Jitter. Ein weiterer Parameter ist *timesync*, dieser wird zur Konfiguration des Time Aware Shaper (TAS) benötigt. Da sich die Anforderung nach TAS erst bei der Auswertung der Parameter ergeben kann, muss die Zeitsynchronisation nachträglich konfiguriert werden, falls diese nicht von Anfang an gesetzt ist.

Im folgenden Aktivitätsdiagramm wird grafisch veranschaulicht, wie der Netzwerkmanager anhand der Parameter entscheidet, ob eine Applikation im System laufen kann. Der erste Schritt ist die Prüfung der *maxLatency*. Die Latenz darf nicht kleiner als 100µs sein, da dies physikalisch nur bei einer Punkt-zu-Punkt Verbindung möglich ist . Diese bedingt aber keine Konfiguration des Netzwerkes durch den Manager, da dies statisch konfiguriert ist. Als nächstes muss die generelle Zeitrelevanz einer Anwendung geprüft werden. Ist die *maxLatency* größer als 200ms, kann die Applikation als zeitunrelevant betrachtet werden. Resultierend daraus, wird lediglich eine Leitung gesucht, bei der das Ankommen der Daten sicherstellt ist.

Ist Zeitrelevanz jedoch nötig, wird als nächstes geprüft, ob ein Jitter angegeben ist. Dies wird über die *minLatency* > 0 geregelt. Ist diese Null, kann bereits das aktuelle System betrachtet werden und es wird gegebenenfalls schon eine Bestätigung zurückgegeben. Falls das aktuelle Netzwerk nicht ausreichend ist, weil beispielsweise die Latenz nicht eingehalten werden kann, muss es neu konfiguriert werden. So kann eine Latenz von kleiner 2ms den Credit Based Shaper zur Folge haben. Ist eine *minLatency* jedoch gesetzt, bedingt dies den Einsatz eines Time Aware Shaper. Welcher wiederum Zeitsynchronisation bedingt.

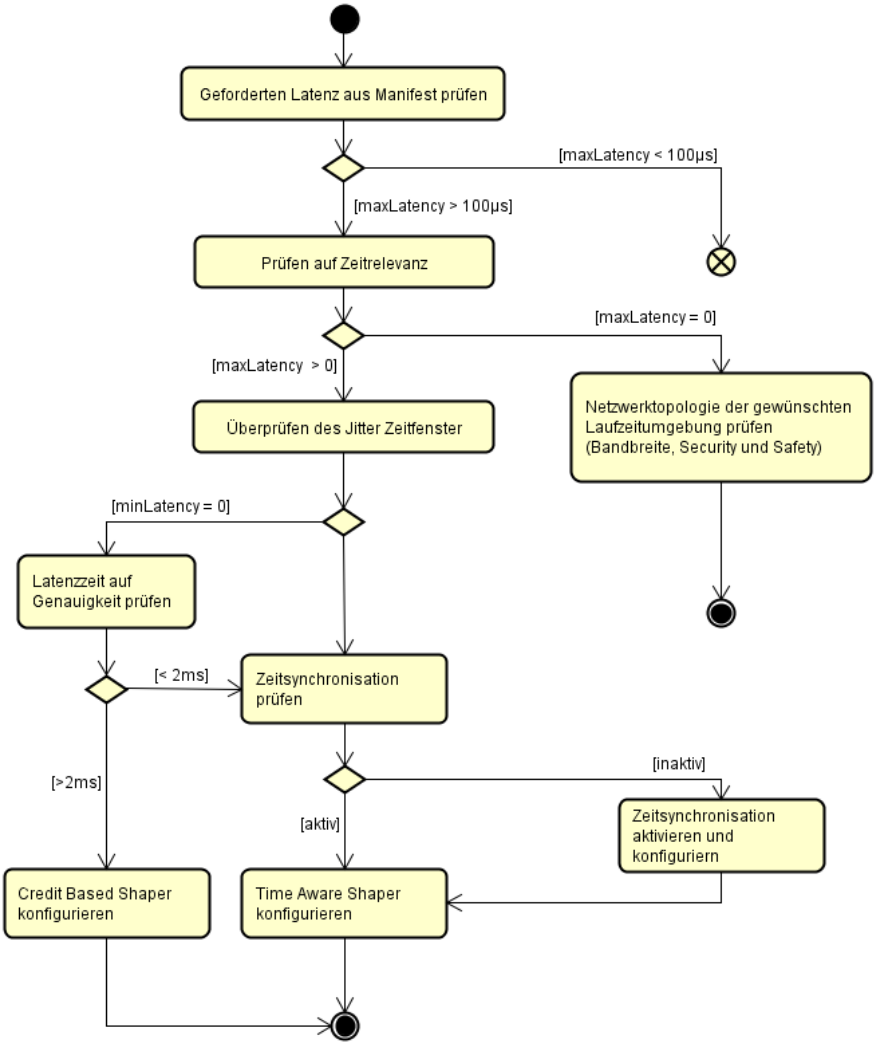

Abbildung 11: Aktivitätsdiagramm Echtzeitfähigkeit

4.3.3 Feature Datenrate

Die bereits in Abschnitt 3.3.4 erwähnte Datenrate besteht aus zwei Parametern. Das folgende Aktivitätsdiagramm zeigt den Ablauf einer Prüfung des aktuellen Netzwerkes. Diese wird anhand der Datenrate aller laufenden Prozesse durchgeführt. Reicht die restliche Bandbreite nicht mehr aus, wird eine Rekonfiguration des Netzwerkes versucht. Erst wenn diese nicht ausreichend ist, wird die Anwendung abgelehnt. In den meisten Fällen wird eine Leitung nie vollständig ausgelastet, um im Falle von *Bursts* keinen Datenverlust zu haben. Als *Bursts* werden Schwankungen in der Datenrate bezeichnet. Dies können zu einer Überlastung der Leitung führen.

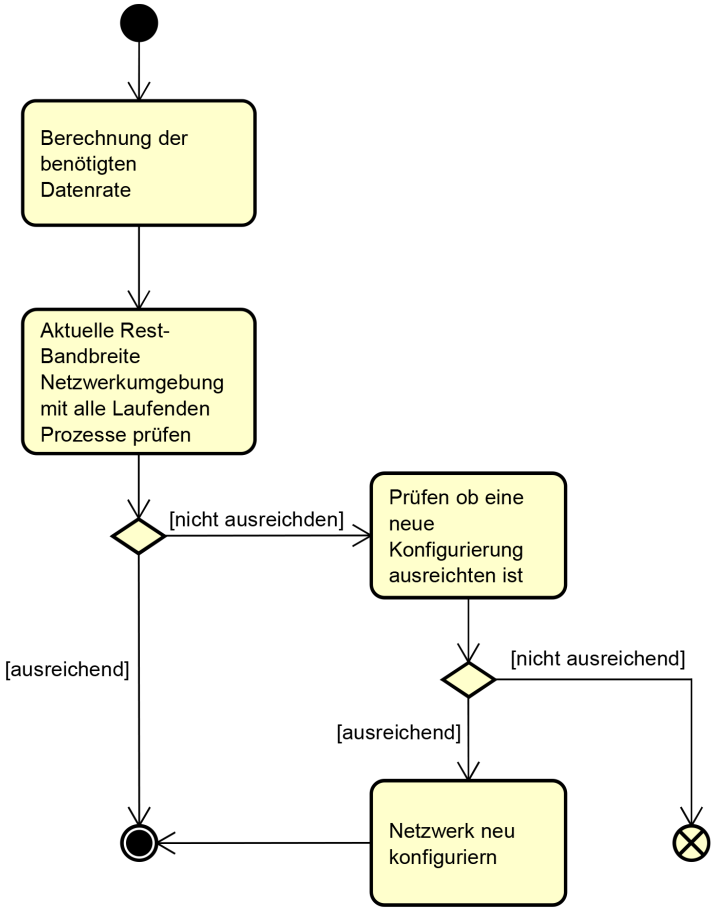

Abbildung 12: Aktivitätsdiagramm Datenrate

4.3.4 Feature Zeitsynchronisation

Die in Abschnitt 3.3.6 erwähnte Zeitsynchronisation ist nicht nur für die Echtzeitfähigkeit wichtig, sonder auch für einen synchronisierten Datenaustausch. Um sicher zu stellen, dass Daten aktuell sind, wird beispielsweise ein gemeinsamer Zeitstempel verwendet. Das Feature Zeitsynchronisation verbindet die Applikation mit einer Master-Clock im System. Ist keine passende Uhr vorhanden, können Mechanismen wie der Best-Master-Clock-Algorithmus angewendet werden [20]. Dieser sucht im System die genaueste Uhr und setzt dies als Master-Clock. Im aktuellen Netzwerk ist die jedoch nur als Backup im Falle eines Ausfalls einer Master-Clock gedacht, da dies momentan manuell vordefiniert ist.

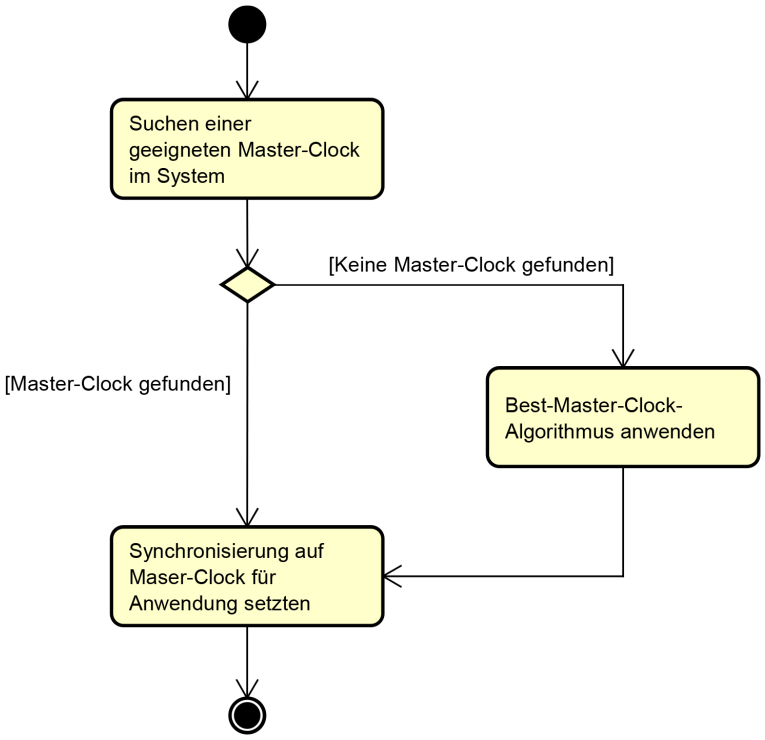

Abbildung 13: Aktivitätsdiagramm Zeitsynchronisation

4.3.5 Feature Integrität

Das Feature Integrität soll eine Überwachung kritischer Daten ermöglichen. Im Manifest ist es als ein Flag modelliert. Der Wert in *integrity* entspricht einem boolischen Wert im Bereich 0 und 1. Dies ist gleichzusetzen mit einem ein- und ausschalten der Funktion. Falls das Flag duch eine 1 gesetzt ist, wird im aktuellen System die Technologie MACsec (Abschnitt 2.2.2) zu konfigurieren. MACsec wir an bei allen Teilnehmern der Kommunikationsstrecke an den jeweils verwendeten Port konfiguriert. Es gibt jedoch Endgeräte die die Technologie noch nicht unterstützen, dies muss bei der Konfiguration beachtet werden. Im nachfolgen Diagramm wird der Prozesses für alle beteiligten Netzwerkgeräte dargestellt.

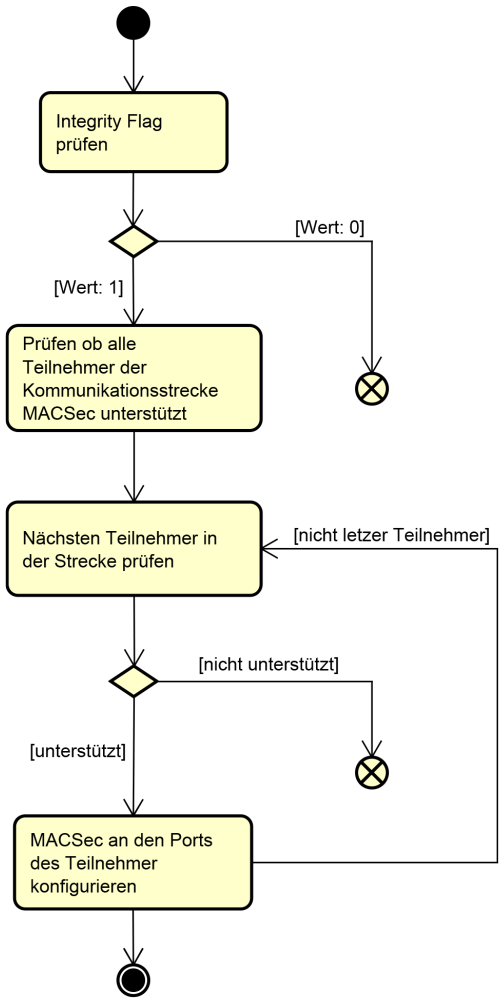

Abbildung 14: Aktivitätsdiagramm Integrität

Kapitel

5

Implementierung der Features

Das Ergebnis aus der dem Entwurf des Maifestes wird eine erste Implementation eines Testszenarios sein. Dies ist ebenfalls Bestandteil dieser Arbeit und wird mittels einiger der in Kapitel 4.3 definierten Features umgesetzt. Da diese Arbeit inhaltlich und zeitlich begrenzt ist, werden nicht alle genanten Features umgesetzt, sondern lediglich ein Szenario im System, welches für später Tests bereits physikalisch getestet werden kann. Dies wird mittels eines Demonstrators, welche bereits im A3F Projekt existiert, durchgeführt. Alle hierfür benötigten Features werden in der Arbeit umgesetzt und dokumentiert.

Einen generellen Überblick aller Relevanten Komponenten und deren Kommunikationsschnittstellen, sog. Interfaces, werden in Folgenden Komponentendiagramm kurz dargestellt. Dies soll die Verbindungen zwischen den Komponenten, sowie die Vernetzung innerhalb der Unterkomponenten vereinfacht darstellen.

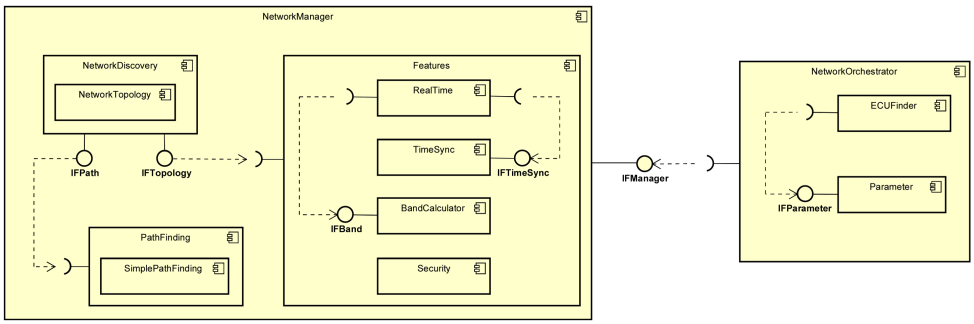

Abbildung 15: Komponentendiagramm Netzwerkmanager

Da besonders im Bereich autonomes Fahren ein hoher Anspruch an die Ausfallsicherheit für eine Anwendungen im System gilt, werden im Nachgang noch mehrere Unit-Tests durchgeführt. Anhand dieser kann die Software auf verschiedene Eingabefälle geprüft werden. In Unit-Test wird mittels definierter Eingabe versucht eine zuvor festgelegte Ausgabe zu erzeugen. Nur wenn dieses Ergebnis mit der gewünschten Ausgabe übereinstimmt, wird der Test als bestanden gewertet. Dies garantiert eine möglichst vielschichtige Abdeckung an Szenarien, die im Betrieb auftreten können.

5.1 Testfall Ingress Filtering

Der gewählte Testfall bezieht sich auf das Ingress-Filtering eines Datenstroms. Dies wird an den jeweiligen Ports, der sich in der Kommunikationsstrecke befindenden Geräten, konfiguriert. Es handelt sich um eine Filterung der nicht erlaubten Teilnehmer. Ingress-Filterring ist ähnlich aufgebaut wie eine Firewall. Im Initialzustand sind alle Kommunikationen verboten. Erst wenn Regeln definiert werden, die das ein- oder ausgehen von Nachrichten erlauben, kann eine Kommunikationsstrecke aufgebaut werden. So werden bei einer Ende-zu-Ende Kommunikation lediglich die beiden Endpunkte zugelassen, die auch in den Regeln festgelegt sind. Hierbei werden die Partner anhand ihrer MAC-Adressen in den Regeln definiert. So wird sichergestellt, dass nur die gewünschten Teilnehmer die Datenpakete empfangen und versenden können.

Für die Regelung der Kommunikation haben sich hauptsächlich zwei Verfahren etabliert.

- *Application Whitelisting*
  Dieses Verfahren verbietet zuerst jegliche Kommunikation. Es müssen explizite Regeln für den Nachrichtenaustausch angelegt werden, um diese zu erlauben. In einem Netzwerk mit vielen Teilnehmern, bei denen nur wenige untereinander eine Verbindung aufbauen dürfen, ist dies ein vielversprechender Ansatz. Es werden wenige Regel benötigt, da Verbote nicht definiert werden müssen.

- *Application Blacklisting*
  Beim *Blacklisting* ist grundsätzlich jegliche Verbindung zugelassen. Ein Verbot der Kommunikation zweier Partner kann nur durch eine Regel definiert werden. Dieser Ansatz macht nur Sinn, wenn es sehr viel Netzwerkteilnehmer gibt, bei denen die meisten untereinander Kommunizieren dürfen. Es müssen somit nur wenige Regeln für Verbote definiert werden.

Für den Automobil-Bereich hat sich der erste Ansatz durchgesetzt, da es eine überschaubare Anzahl an Kommunikationspartnern gibt, die wiederum aus Sicherheitsgründen nicht alle untereinander kommunizieren dürfen. Ebenso wird durch das explizite Erlauben der Kommunikation das Fehlerrisiko einer Verfälschung der Daten minimiert. Dies kann durch einen anderen Netzwerkteilnehmer erfolgen, der die Nachricht auch empfangen konnte und sie abgefälscht weiterleitet. Ein Verfälschen der Nachrichten kann sowohl mutwillig durch eine manipulierten Netzwerkteilnehmer, als auch unwissentlich durch einen Defekt erfolgen. Beide Fälle müssen im System abgefangen werden, da durch sie ein Risiko entsteht.

5.2 Konfiguration der Endgeräte

Die Endkonfiguration die in automatisch im Netzwerkmanager erstellt wurde, wird mit einem eigens entwickelten Konfigurator auf die Endgeräte gespielt. So ist es möglich unabhängig vom Gerätetyp eine Konfiguration vor zu nehmen. Der Konfigurator kennt die möglichen Einstellungen an jeder ECU oder jedem Switch. Um die endgültige Konfiguration vor zu nehmen, gibt es zwei Ansätze. Der erste ist eine statische Konfigurationsdatei im Extensible Markup Language (XML) Format an das Gerät zu übermitteln, in dem alle Einstellungen hinterlegt sind. Die zweite, weitaus agilere Methode ist es, mittels Funktionsaufrufen im Konfigurator die Einstellungen je nach bedarf am Endgerät zu ändern. Dies hat den Vorteil, dass pro Feature die jeweiligen Einstellungen abgeschlossen werden könnten und nicht erst in eine XML Datei gespeichert werden müssen.

5.3 Umsetzung der Parametereinlesung

Wie in Kapitel 4 bereits erwähnt, bekommt der Netzwerkmanager seine Parameter mittels JSON File übermittelt. Dies muss im ersten Schritt ausgelesen werden, und den entsprechenden Variablen im System zugeordnet werden, um im Nachgang damit zu arbeiten. Hierzu wurde die Klassen *json_translator* und *in_car_application* erstellt. Die erst ermöglicht die Konvertierung der Parameter in die benötigten Variablentypen, welche in *in_car_application* definiert sind. Sie ist jedoch auch in der Lage zu erkennen ob eine Anwendung mehrere Nachrichten versenden will und berechnet die Gesamtbandbreite der Applikation. Dies ist wichtig, damit der Netzwerkmanager später entscheiden kann, ob noch genügend Bandbreite vorhanden ist.

Beim Aufruf der Metode zum hinzufügen einer neuen Anwendung, wird ein Objekt vom Typ *InCarApplication* zurück gegeben, in diese sind alle eingelesen Parameter die applikationsabhängig sind enthalten. Eine Applikation wird in der Klasse *in_car_application* abgebildet.

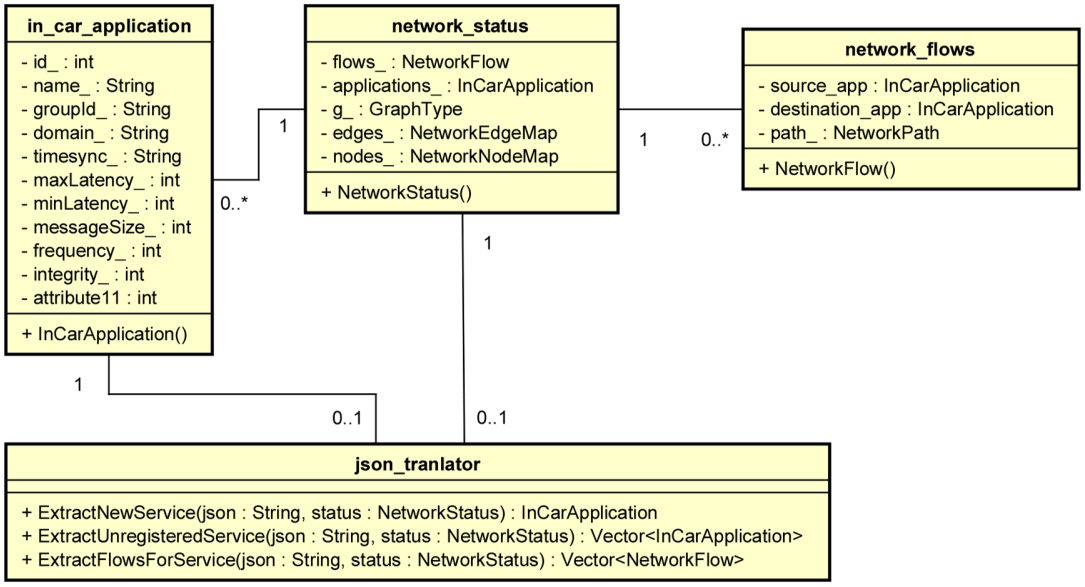

Abbildung 16: Klassendiagramm Parametereinlesung

Des weiteren erhält der Netzwerkmanager noch den sog. Netzwerkstatus. In diesem ist der aktuelle Stand des Netzwerkes abgelegt. So sind dort beispielsweise Flows(Routen), Laufende Anwendungen und einige weitere Parameter enthalten, welche aber für die weiter Implementation in diesem Fall nicht von Bedeutung sind. Der Zusammenhang zwischen den verschieden Klassen die für die Parametereinlesung verantwortlich sind, ist im obigen Klassendiagramm 16 dargestellt. Ein Aufruf von außen, beispielsweise durch den Orchestrator, wird mittels der in Abschnitt 4.2.1 REST Schnittstelle realisiert. Mit dieser werden alle benötigten Daten zwischen den Schnittstellen ausgetauscht.

5.4 Umsetzung der Parameterauswertung

Nachdem die Parameter in den entsprechenden Variablen hinterlegt wurden, beginnt der Netzwerkmanager nacheinander die Features aus Kapitel 4.3 abarbeiten um die richtige Konfiguration zu finden. In dem Testfall Ingress Filtering wird mit dem Aufbau der Kommunikationsstrecke begonnen. Dazu werden die vom Orchestrator übergeben ECU's als Start- und Endknoten verwendet. Mithilfe der Klasse *simple_path_resolution* werden mögliche Pfade zwischen den Endpunkten gesucht. Dies müssten im Anschluss anhand der

weiteren Features Echtzeitfähigkeit, Datenrate, Zeitsynchronisation und Integrität untersucht werden. Im Testfall geht es jedoch nur um den Aufbau einer Kommunikationsstrecke mit den dazugehörigen Portregelungen für alle Teilnehmer. In Abbildung 17 werden die benötigten Klassen grafisch dargestellt. In der Klasse *feature_based_configuration_resolution* ist für die Aufrufe der einzelnen Features aus Kapitel 4.3 verantwortlich. Um einen einheitlichen Aufbau und Aufruf der Features sicher zu stellen wurde ein Interface definiert, welches die nötigen Funktionalität eines Features fest legt. Die Ergebnisse der Funktionen werden in einer Datenstruktur namens *InstructionMapping*, welches eine eine Liste der errechneten Ergebnisse und Einstellungen aus den Featuren enthält.

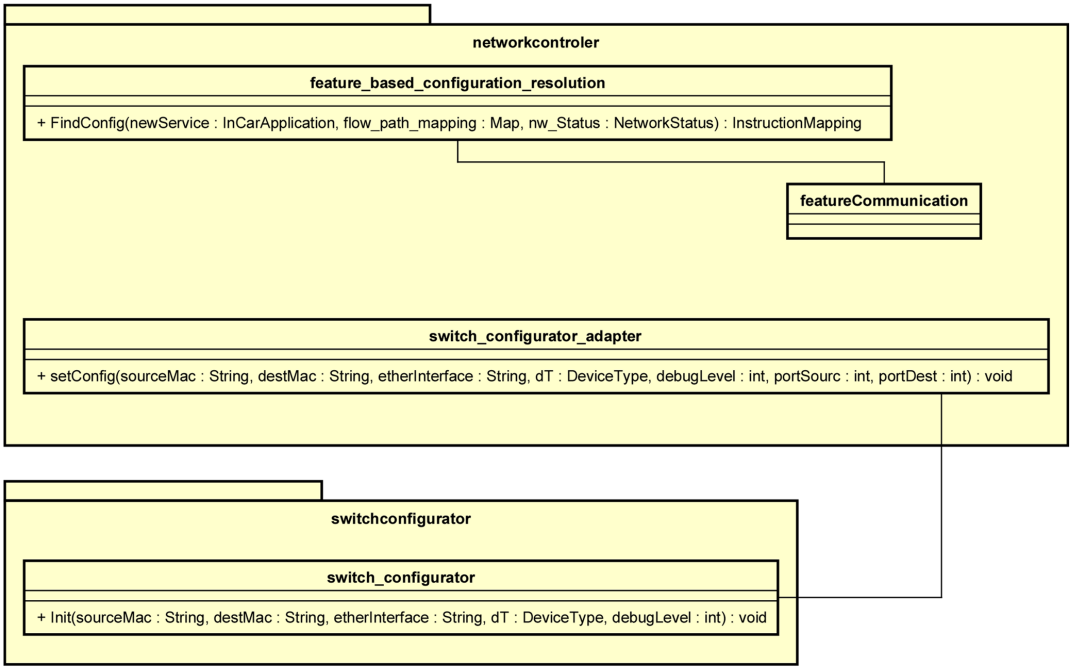

Abbildung 17: Klassendiagramm Switch-Konfiguration

5.5 Umsetzung der Konfiguration des Endgerätes

Abbildungsverzeichnis

Literaturverzeichnis

[1] *Automatisiertes Fahren*. https://www.vda.de/de/themen/innovation-und-technik/automatisiertes-fahren/automatisiertes-fahren.html. Abgerufen: 07.06.2018.

[2] *Dan Des Ruisseaux. Designing a Deterministic Ethernet Network (Whitepaper): Industrial Communication. Hrsg. von Schneider Electric Industries SAS*. http://www.schneider-electric.co.uk/documents/solutions/process-automation/open-connectivity/Designing%20a%20Deterministic%20Ethernet%20Network.PDF. Abgerufen: 23.05.2018.

[3] *ECMA-404 The JSON Data Interchange Standard*. https://www.json.org/json-de.html. Abgerufen: 20.04.2018.

[4] *Heterogenes Netzwerk*. https://www.itwissen.info/Heterogenes-Netzwerk-heterogeneous-network.html. Abgerufen: 26.06.2018.

[5] *IEEE 802 Numbers*. https://www.iana.org/assignments/ieee-802-numbers/ieee-802-numbers.xhtml. Abgerufen: 30.05.2018.

[6] *IEEE 802.1*. http://www.ieee802.org/1/. Abgerufen: 10.04.2018.

[7] *IEEE 802.1AS-Rev*. https://1.ieee802.org/tsn/802-1as-rev/. Abgerufen: 16.04.2018.

[8] *IEEE 802.1CB*. https://1.ieee802.org/tsn/802-1cb/. Abgerufen: 16.04.2018.

[9] *IEEE 802.1CM*. https://1.ieee802.org/tsn/802-1cm/. Abgerufen: 16.04.2018.

[10] *IEEE 802.1Qbu*. http://www.ieee802.org/1/pages/802.1bu.html. Abgerufen: 16.04.2018.

[11] *IEEE 802.1Qca*. http://www.ieee802.org/1/pages/802.1ca.html. Abgerufen: 16.04.2018.

[12] *IEEE 802.1Qcc*. https://1.ieee802.org/tsn/802-1qcc/. Abgerufen: 16.04.2018.

[13] *IEEE 802.1Qci*. https://1.ieee802.org/tsn/802-1qci/. Abgerufen: 16.04.2018.

[14] *IEEE 802.1Qcr*. https://1.ieee802.org/tsn/802-1qcr/. Abgerufen: 16.04.2018.

[15] *IEEE 802.3*. http://www.ieee802.org/3/. Abgerufen: 09.04.2018.

[16] *IEEE Standard for Local and metropolitan area networks: Media Access Control (MAC) Security*. http://www.ieee802.org/1/files/public/docs2010/new-seaman-1AE-markup-for-gcm-aes-256-0710-v2.pdf. Abgerufen: 30.05.2018.

[17] *Kernel-based Architecture for Safety-critical Control*. https://pdfs.semanticscholar.org/9e71/081cbdddc54f83bdaedb743113f04b7d223b.pdf?_ga=2.225926232.335041009.1531333031-98898336.1531333031. Abgerufen: 13.06.2018.

[18] *Precision clock synchronization protocol for networked measurement and control systems*. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4839002. Abgerufen: 07.06.2018.

[19] *Road vehicles — Functional safety*. https://www.iso.org/obp/ui/#iso:std:iso:26262:-1:ed-1:v1:en. Abgerufen: 11.07.2018.

[20] *Specialization of IEEE 1588 Best Master Clock Algorithm to 802.1AS*. http://www.ieee802.org/1/files/public/docs2006/as-garner-bmc-060606.pdf. Abgerufen: 04.07.2018.

[21] Doglio, Fernando: *Pro REST API Development with Node.js*. Springer-Verlag, Berlin Heidelberg, 1. Auflage, 2015, ISBN 978-1-4842-0918-9.

[22] Eckert, Claudia: *IT-Sicherheit: Konzepte - Verfahren - Protokolle*. Walter de Gruyter GmbH & Co KG, Berlin, 9. Auflage, 2014, ISBN 978-348-685-916-4.

[23] James F. Kurose, Keith W. Ross: *Computer networking: A top-down approach*. Pearson Education, Inc, New Jersey, 6. Auflage, 2013, ISBN 978-0-13-285620-1.

[24] Meyer, Gereon: *Advanced Microsystems for Automotive Applications 2012*. Springer-Verlag, Berlin Heidelberg, 6. Auflage, 2012, ISBN 978-3-642-29672-7.

[25] Paul Goransson, Chuck Black, Timothy Culver: *Software defined networks: A comprehensive approach*. MA: Morgan Kaufmann, Cambridge, 2. Auflage, 2017, ISBN 978-012-804-555-8.

[26] Peter Mandl, Andreas Bakomenko, Johannes Weiß: *Grundkurs Datenkommunikation*. Vieweg+Teubner GWV Fachverlage GmbH, Wiesbaden, 1. Auflage, 2008, ISBN 978-3-8348-0517-1.

[27] R. Ramaswamy, Ning Weng, T. Wolf.: *Characterizing network processing delay*. IEEE Global Telecommunications Conference, 2004, GLOBECOM '04. IEEE, 1. Auflage, 2004, ISBN 0-7803-8794-5.

[28] Tanenbaum, Andrew S.: *Computer Network*. Pearson Education, Boston, 5. Auflage, 2011, ISBN 978-0-13-212695-3.