INSTITUT FÜR INFORMATIK

Fehlertoleranz bei Prozessabläufen - mit Anwendungen bei akustischen Unterwassernetzwerken

Fatih Bülbül, Thies Petersen, Michael Recker,
Fabian Sell, Kim Wachlin
Betreuer: Ivor Nissen

Bericht Nr. 1703
Mai 2017
ISSN 2192-6247

CHRISTIAN-ALBRECHTS-UNIVERSITÄT

ZU KIEL

Institut für Informatik der
Christian-Albrechts-Universität zu Kiel
Olshausenstr. 40
D – 24098 Kiel

Fehlertoleranz bei Prozessabläufen - mit Anwendungen bei akustischen Unterwassernetzwerken

Fatih Bülbül, Thies Petersen, Michael Recker,
Fabian Sell, Kim Wachlin
Betreuer: Ivor Nissen

Bericht Nr. 1703
Mai 2017
ISSN 2192-6247

e-mail: in@is.informatik.uni-kiel.de

Beiträge zur Kooperation autonomer Teams unter Wasser.
Dieser Bericht enthält die Dokumentation der Gemeinschaftsarbeit des Bachelorpraktikums der Verfasser

Inhaltsverzeichnis

1 Vorwort

Ivor Nissen

Es gibt einige Vorhaben, die sich mit Robotorschwärmen in Ozeanen beschäftigen, ohne dass die unteren Ebenen von Kooperation, Netzwerk und Bitübertragung befriedigend gelöst sind. Ein Schwarm von Subjekten ist damit schwerlich realisierbar. In diesem Beitrag sollen für autonome Teams erste Basisfähigkeiten eines Netzwerkes aufgezeigt und behandelt werden.

Die robuste digitale Unterwasserkommunikation im Verbund, beispielsweise im mobilen Ad-hoc-Netzwerk (MANET) in der Wassersäule, ist erst seit weniger als zwei Jahrzehnten technisch möglich und Forschung auf diesem Gebiet wird weltweit konsequent vorangetrieben. Mit der Kommunikation in Netzwerken entstehen auch Anwendungen, die auf Kooperation, Koordinierung und Koalition aufbauen, beispielsweise für autonome Roboterteams. Einige Problemlösungen benötigen zwingend diese Form der Interaktion. Damit entstehen verteilte Systeme mit dezentralen Fehlerquellen. Das Ziel sind fehlertolerante und damit verlässliche Netzwerke. Doch wie können Prozesse hierfür entwickelt und validiert werden? Wann ist das MANET verlässlich?

Ein Beispiel: Es ist nicht absehbar, dass aufgrund der anspruchsvollen, fluktuierenden akustischen Übertragungsbedingungen vollständige SONAR-Bilder mittels Unterwasserkommunikation ausgetauscht werden können. Die linke der folgenden Abbildung stellt eine bereits stark komprimierte Orginalaufnahme dar, bei der sich ein Bodenknoten mit Schattenwurf (Pelle mit Modem) in dem oberen rechten Quadranten links der Mitte abzeichnet. Die zugehörige Bilddatei beansprucht eine Größe von 651804 Bytes. Eine Komplettübertragung über wenige Kilometer bei realistischen 300 bps würde knapp fünf Stunden benötigen und sicherlich vorher fehlerhaft abbrechen, da die Energieressourcen des autonomen Fahrzeugs verbraucht worden sind. Zudem würde die Kommunikation das SONAR teilweise stören. Für eine Objektidentifizierung durch den Operator könnte eine Kooperation hilfreich sein und zum gleichen Ziel führen. Eine grobe Übersichtsübertragung mit einer Größe von 30054 Bytes beispielsweise in Form vom untersten JPEG

2000-Layer würde in knapp 13 Minuten abgeschlossen sein (rechtes Bild). Der Anwender könnte dann in einem zweiten Schritt einen Bildausschnitt gleicher Dateigröße aussuchen und eine Zusendung anfordern. Diese Interaktion spart 90% des Aufwandes ein, benötigt jedoch Interoperabilität und definierte Prozessabläufe (Mini-Stories).

Das gleiche Prinzip könnte beim interaktiven Auslesen einer Black-Box am Meeresboden angewendet werden.

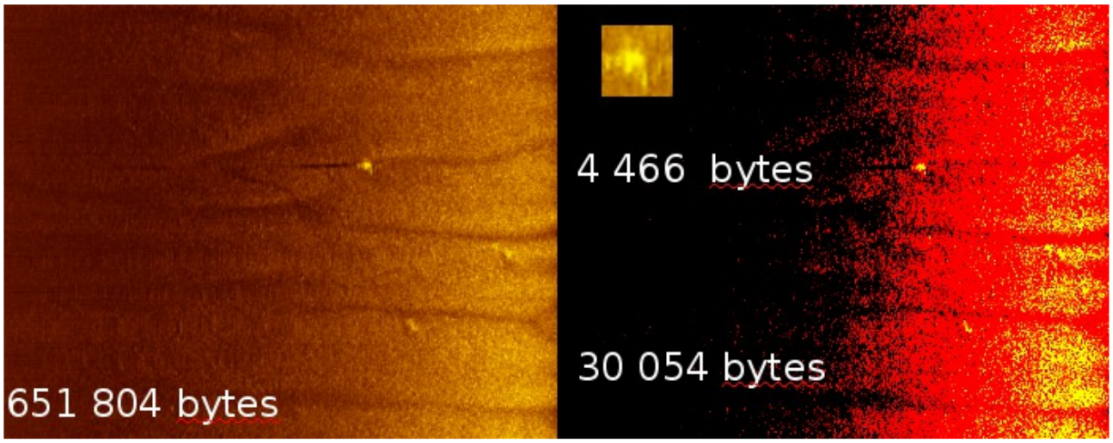


Abbildung: *Links SONAR-Aufnahme mit einem Bodenobjekt links-mittig im oberen rechten Quadranten. Rechts: unterste Ebene des hierarchischen JPEG2000-Bildformates und oben ein Ausschnitt des Bodenobjektes ohne Schattenwurf in der höchsten Auflösungsstufe der JPEG-Pyramide.*

Derzeit existieren diese Prozessabläufe jedoch nicht. Es gibt keinen Kommunikationsstandard für Unterwassersensornetzwerke; es wird jedoch beispielsweise an dem internationalen Standard ISO/IEC 30140 (Underwater acoustic sensor network (UWASN)) gearbeitet. Seit der FWG-Forschungsfahrt „CCUP 2007" mit Beteiligung des damaligen NATO Underwater Research Centers (NURC, jetzt CMRE) im Mittelmeer, wird an einem Erst-Kontakt-Protokoll mit der Bezeichnung JANUS (ANEP 87) als Standard-Kommunikationsstack (komplett von der Bitübertragung bis zur Anwendung) als „maschineller Channel 16" für den Unterwasserbereich gearbeitet. Nachdem in den JANUS-Entstehungsjahren 2008 und 2009 eine einheitliche Anwendung (all-in-one-solution) international nicht umsetzbar war - man konnte sich nicht auf die Belegung von 9 freien, noch nicht belegten Bit einigen - wurde in 2010 durch den Betreuer dieser Gruppenarbeit ein Multi-Anwendungskonzept mit unterschiedlichen Klassenanwendung (user classes) und Anwendungsschablonen vorgestellt. Dieser Zugang erlaubt es, dass viele Anwender ihre

eigenen individuellen und öffentlichen Anwendungen als sogenannte application types spezifizieren können. Er wurde 2010 international in der Version 2 angenommen und für die JANUS-Version 3 in dem Ratifizierungsprozess übernommen. Danach gibt es offene, allgemein zugängliche Anwendungen wie „Emergency", „Underwater GPS", „Underwater AIS", Pinger (ranging), „First Contact", „Keep silent!-EMCON-Beacon", Kommunikation mit Offshore-Infrastruktur („Wind and Wave Power Generator"), nachrichtentragende Beacon (Fixed Vertical Mooring, Hazard Marker) und nationale respektive properitäre Anwendungen. Jedoch sind diese Anwendungsschablonen noch nicht mit Leben gefüllt, durchgeführte Studien bisher international unakzeptiert.

- Welche Nachrichteninhalte und welche Parameter sind zwingend zu übertragen?
- Welche Interaktionen müssen folgen und was passiert, wenn eine Antwort ausbleibt?
- Was ist nun, wenn in diesem verteilten System Fehler auftreten, wie im obigen Fall unzulässige Bildausschnittskoordinaten auftauchen?
- ...

Niemand hat in dem verteilten System eine globale Sicht, es gibt nicht den einen gemeinsamen Zeitpunkt. Wie ist zu einem bestimmten Zeitpunkt der Energievorrat in den autonomen Systemen? Gibt es einen globalen Zustand des MANET? Da mittels FEC-Techniken Übertragungsfehler minimiert (Forward Error Correction) und mit Checksummen syntaktische Fehler auf der Ebene des Netzwerkes ausgeschlossen werden können, mit welchen Fehlertypen muss gerechnet werden? Es sind Mutationen in Form von semantisch inkorrekten Nachrichten wie Positions- und Zeitstempelfehleintragungen aber auch Kommunikationsknotenabstürze (Wassereinbruch, Energiemangel), Timing-Fehler, Antwortzeiten außerhalb von Spezifikationen und vieles mehr anzutreffen. Zudem sind auch bewusste Manipulationen (Spoofing) in Form von zufälligen Antworten zu jeder Zeit (Byzantinische Fehler) nicht auszuschließen.

In diesem Wirtschaftsinformatiker-Praktikum der Arbeitsgrupppe Technologie der Informationssysteme an der Universität Kiel werden in diesem Beitrag erste Basis-Kooperationen vorgestellt, die überwiegend JANUS nutzen. Zur Modellierung und zum Test der Prozessabläufe wird S-BPM (Subject-oriented business process management) in der Java-Suite Metasonic herangezogen. Hierdurch können Deadlocks gefunden und analysiert werden.

Diese Arbeiten sollen auf die Problemstellung hinführen und Bachelorarbeiten im Bereich der Fehlertoleranz mit den bekannten Multihop-Herausforderungen (Zwei-Armeen- und Byzantinisches Generäle-Problem) durch die Entwicklung eines Anwendungs-Simulations-Frameworks vorbereiten. Beispielsweise findet sich der Drei-Wege-Handschlag beim „Mark-Snap-Verfahren" als Vertreter der IM-RES-Verfahren im Kapitel 5 wieder. Ein Konzept der Dolmetscher-Fähigkeit wird vorgeschlagen (Kapitel 4), autonome Auftragsumgestaltungen der Maschinen tangieren den Bereich vom „Lernenden MANET" (Kapitel 7).

Ivor Nissen

Danksagung: Essentiell für die Arbeiten zu den Prozessabläufen ist eine adäquate und verlässliche Modellierung der S-BPM-Prozesse. Zur Unterstützung dieser Studentenarbeit wurden von der Firma Allgeier IT Solutions GmbH für die Software Metasonic Suite temporäre Lizenzen zur Verfügung gestellt, mit der auch die dargestellten Prozess-Graphiken generiert worden sind. Wir danken Tim Waldberg, Head of Metasonic Consulting, für seine unbürokratische Hilfe und damit Machbarkeit dieses Beitrages.

2 Einleitung

2.1 Grundlagen mit Kommunikationsprozessen

In dieser Arbeit setzen wir uns mit der Modellierung von Kommunikationsprozessen im Hinblick auf ihre Fehlertoleranz auseinander. Als Anwendungsfall wurde dabei die Kommunikation von verteilten, teilweise mobilen Knoten, unter Wasser gewählt. Fehlertolerante Kommunikationsprozesse werden allerdings auch in vielen anderen Bereichen benötigt. Exemplarisch seien hier der Straßenverkehr mit autonomen Fahrzeugen oder auch die Logistik genannt. Viele in dieser Arbeit ausgearbeitete Prozesse können auch auf anderen Anwendungsfälle übertragen werden.

Die Arbeit gliedert sich in eine Einleitung, in der das verwendete Tool der Metasonic Suite zur Modellierung von S-BPM erläutert sowie in die Fehlertoleranzproblematik eingeführt wird. Daran anschließend stellen wir einen Prozess zur Nachrichtenübermittlung mittels Flooding und den verwendeten Aufbau der ausgetauschten Nachrichten vor. Im folgenden Kapitel wird anhand des JANUS Protokolls ein Prozess erarbeitet, wie Knoten untereinander den Kontakt herstellen und die Kommunikation aufbauen können.

Ein wichtiger Aspekt der Arbeit ist, wie die beteiligten Subjekte den eigenen Abstand zu anderen Subjekten feststellen können. Eine Modellierung dieses Prozesses findet sich im Kapitel Distance Measurement. Unter Wasser ist die Kommunikation von verteilten Knoten nicht ohne weiteres möglich. In unserer Arbeit benutzen wir die Idee eines Postboten, der Nachrichten von Knoten einsammelt und ausliefert. Im Kapitel Task Handover wird untersucht, wie der Ausfall eines Knotens kompensiert und die Aufgabe an einen anderen übergeben werden kann.

Alle fünf Hauptkapitel dieser Arbeit haben einen ähnlichen Aufbau. Es wird kurz in die benötigte Basiskooperation eingeführt, anschließend werden die im weiteren betrachteten Prozess und mögliche Komplikationen formuliert. Ein Schwerpunkt wird danach auf die jeweils ausgetauschten Nachrichten und die Modellierung der Prozesse in S-BPM gelegt.

Jedes Kapitel endet mit der Betrachtung der Fehlertoleranz.

2.2 MANET - mobile Ad-hoc-Netzwerke

MANET steht für *mobiles Ad-hoc-Netzwerk* und bedeutet, dass verteilte mobile Knoten ein Netzwerk aufbauen, um zu kommunizieren. Dabei ist jeder Knoten nicht nur Sender und Empfänger von Nachrichten sondern auch Relay. Das heißt, ein Knoten reicht empfangene Nachrichten, die nicht für ihn bestimmt sind, an seine Nachbarknoten weiter.
Vorteilhaft ist hierbei, dass der Ausfall eines einzelnen Knoten nicht dazu führt, dass das gesamte Netzwerk zusammenbricht, da die Nachbarknoten die Funktion des ausgefallenen Knoten übernehmen. Weiterhin ist, insbesondere für das in dieser Arbeit behandelte Einsatzgebiet, die hohe Mobilität des MANETs als Vorteil festzuhalten.
Entsprechend nachteilig gegenüber einem Netzwerk mit zentralem *Master* ist, dass jeder Knoten über eine gewisse Grundfunktionalität verfügen muss - insbesondere, um die benötigten Relay-Funktionen zur Verfügung stellen zu können.
Bei einem Einsatz unter Wasser bilden mehrere autonome Unterwasserfahrzeuge als Knoten ein MANET. Insbesondere beim Einsatz unter Wasser müssen - statt der Nutzung bewährter Technologien - neue Wege gefunden, um Nachrichten auszutauschen und die Koordinierung autonomer Teams zu gewährleisten, da die Übertragungsbandbreite stark eingeschränkt ist, die Übertragungsverzögerung im Bereich von Sekunden liegt, viele Pakete nicht oder fehlerbehaftet ankommen und eine dominierende Orts- und Wetterabhängigkeit besteht. Um Kollisionen im Netzwerk zu minimieren, ist zudem die maximale Größe von Nachrichten stark beschränkt.

2.3 Grundlagen des S-BPM-Werkzeuges

2.3.1 Allgemein

S-BPM steht für *Subject-oriented Business Process Management* und ist eine Modellierungsnotation für die Beschreibung von Geschäftsprozessen, wobei die Kommunikation im Vordergrund steht. Diese beschränkt sich dabei auf nur wenige Modellierungssymbole, deren Fokus ebenfalls vorwiegend auf Kommunikation und Nachrichten liegt. Ein wesentliches Merkmal von S-BPM ist, dass aus der Ich-Perspektive des Subjekts modelliert wird und somit nebenläufige Prozesse und deren Nachrichtenaustausch präzise beschrieben werden können. Dies geschieht in der natürlichen Sprache. Weiterhin ist eine direkte Umwandlung des Modells in eine ausführbare Form und damit auch eine Simulation möglich.

2.3.2 Symbolik

Wie bereits erwähnt, bedient sich S-BPM nur weniger Symbole zur Modellierung der Prozesse. Insgesamt besteht es dabei lediglich aus folgenden fünf Symbolen, die in der folgenden Abbildung graphisch dargestellt werden:

- **Subjekt**
  Subjekte sind alle Akteure, also Prozessbeteiligten. In dieser Ausarbeitung werden es zum Beispiel AUVs oder Bojen sein.
- **Nachricht**
  Nachrichten dienen dem Informationsaustausch. Sie werden daher zur Modellierung der Kommunikation zwischen Subjekten verwendet.
- **Sendezustand**
  Dieses Symbol wird benutzt, Wenn ein Subjekt einem anderen .
- **Empfangszustand**
  Analog zum Sendezustand wird der Zustand eines Subjektes modelliert, wenn es die Nachricht empfängt.
- **Interne Funktion**
  Eine interne Funktion wird benutzt, um ein internes Verhalten eines Subjekts (ohne Interaktion mit anderen Subjekten) zu modellieren.

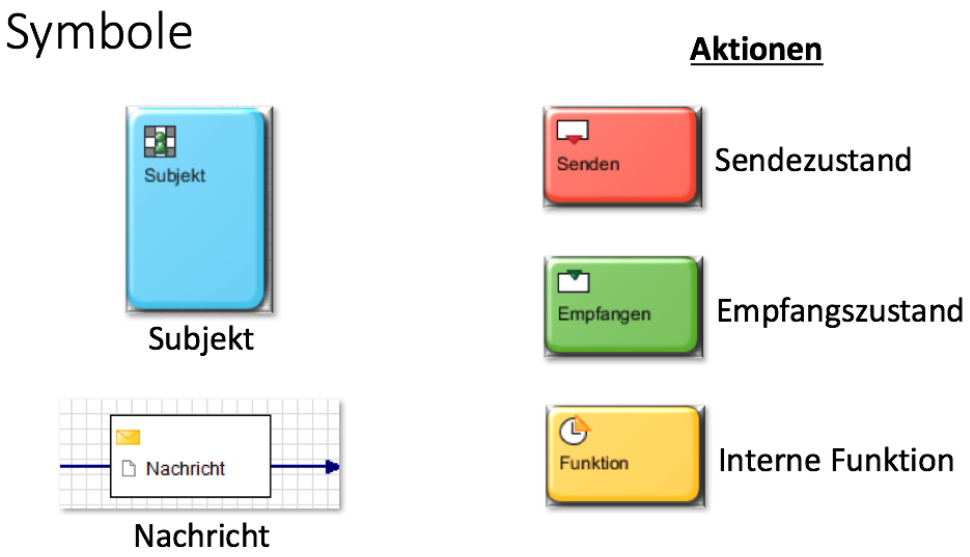


Abbildung 2.1: Symbole zur Modellierung in S-BPM

2.3.3 Modellierung

Generell wird S-BPM in zwei verschiedenen Varianten modelliert, den sogenannten *Ansichten*:

1. **Die Kommunikationssicht**
   Die Kommunikationssicht zeigt den kompletten Prozess als Übersicht. Dabei modelliert man alle beteiligten Subjekte und die Kommunikation zwischen den Subjekten. Dementsprechend werden in dieser Ansicht die Symbole *Subjekt* und *Nachricht* verwendet.

2. **Die interne Subjektansicht**
   Die interne Subjektansicht zeigt die internen Geschehnisse eines Subjekts auf. Dabei modelliert man normale Funktionen als *interne Funktion*, kommunikationsauslösende Ereignisse als *Sendezustand* und Aktionen, bei denen Nachrichten erwartet werden, als *Empfangszustand*.

2.3.4 Beispiel: Pizzabestellung

Damit wir uns besser in die Metasonic Suite einarbeiten und mit dem Thema S-BPM auseinander setzen können, haben wir uns zu Beginn des Projekts dazu entschieden, einen übersichtlichen Prozess des täglichen Lebens zu modellieren. Wir haben hier den

Bestellvorgang für eine Pizza gewählt. Um dem Leser ebenfalls einen einfacheren Einstieg in diesen Modellierungstyp zu gewähren, möchten wir den Prozess hier einmal kurz vorstellen.
Wir steigen erst einmal in die im vorherigen Abschnitte beschriebene Kommunikationssicht ein:

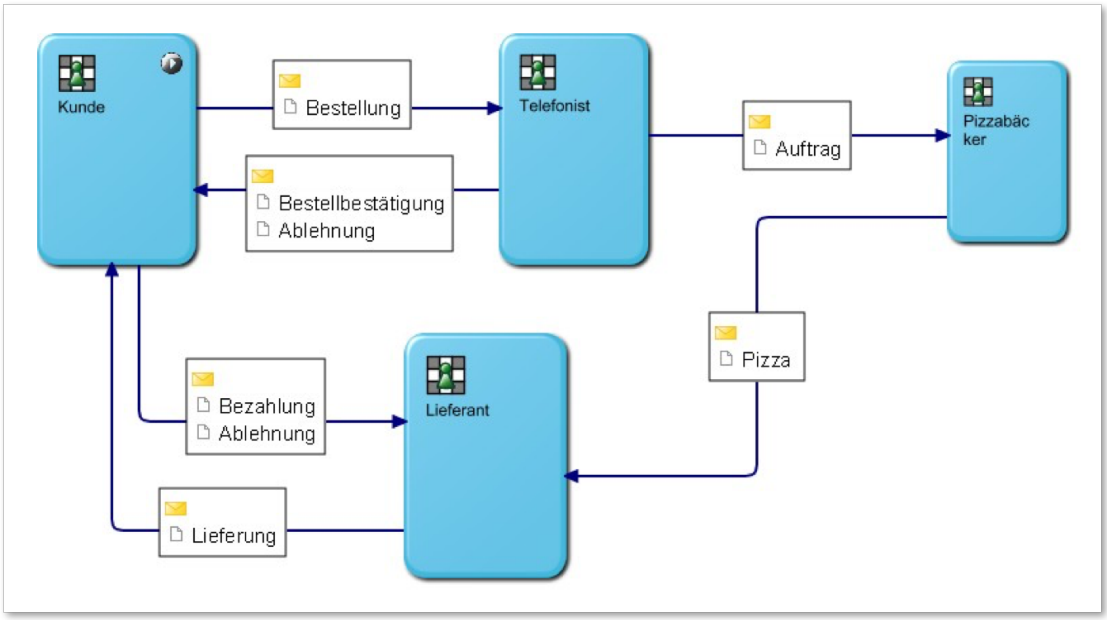


Abbildung 2.2: Kommunikationssicht der Pizzabestellung

In dieser Abbildung sind die verschiedenen Subjekte des Prozesses ersichtlich. Es gibt einen Kunden, den Telefonisten, den Pizzabäcker und einen Lieferanten. Der Kunde hat die Möglichkeit eine Bestellung bei dem Telefonisten abzusetzen und erhält im Gegenzug eine Auftragsbestätigung oder eine Ablehnung, sollte die Bestellung nicht möglich sein. Der Telefonist gibt die Bestellung nach der Auftragsbestätigung an den Pizzabäcker weiter und dieser die Pizza an den Lieferanten. Der Lieferant übergibt die Pizza an den Kunden und bekommt von dem Kunden die Bezahlung oder, wenn diese nicht in Ordnung sein sollte, eine Ablehnung.
Nachfolgenden werden die internen Ansichten der jeweiligen Subjekte erläutert (Abbildung 2.3), beginnend mit dem Kunden:

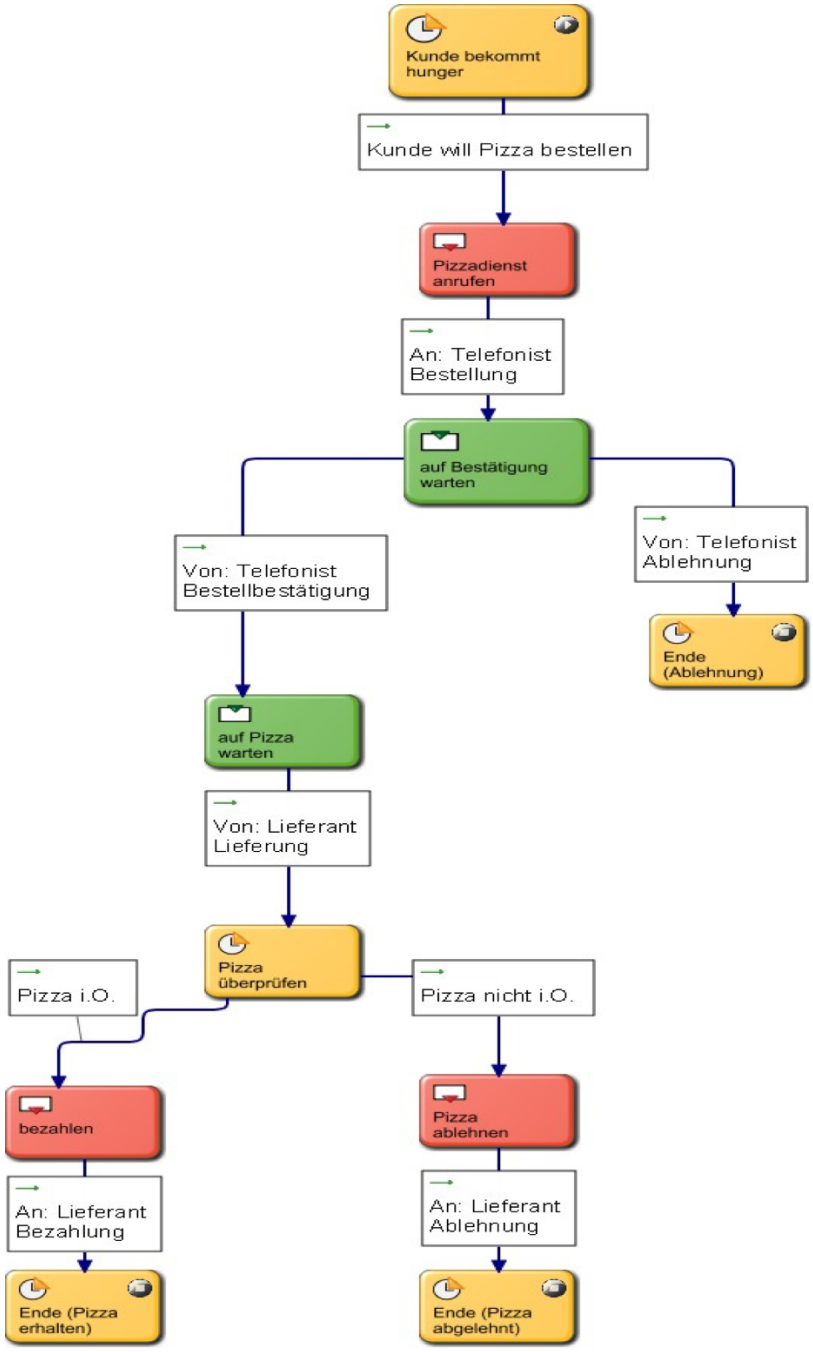


Abbildung 2.3: Interne Ansicht des Kunden

Der Prozess startet mit einem Funktionszustand, in dem der Kunde Hunger bekommt und eine Pizza bestellen möchte. Deshalb ruft er mit einem Sendezustand den Telefonisten an und wartet in dem Empfangszustand auf eine Bestätigung. Im Falle einer Ablehnung ist der Prozess mit dem Funktionszustand „Ende (Ablehnung)“ für den Kunden bereits beendet. Ansonsten erhält der Kunde eine Bestellbestätigung und wartet anschließend auf die Pizza. Wenn der Lieferant die Pizza liefert, kontrolliert der Kunde die Pizza und entscheidet sich dann, ob die Pizza in Ordnung ist oder nicht. Sollte die Pizza in Ordnung sein, dann bezahlt der Kunde den Lieferanten, falls nicht erhält der Lieferant eine Ablehnung. In beiden Fällen ist der interne Prozess des Kunden beendet.

Weiter geht es mit dem internen Verhalten des Telefonisten:

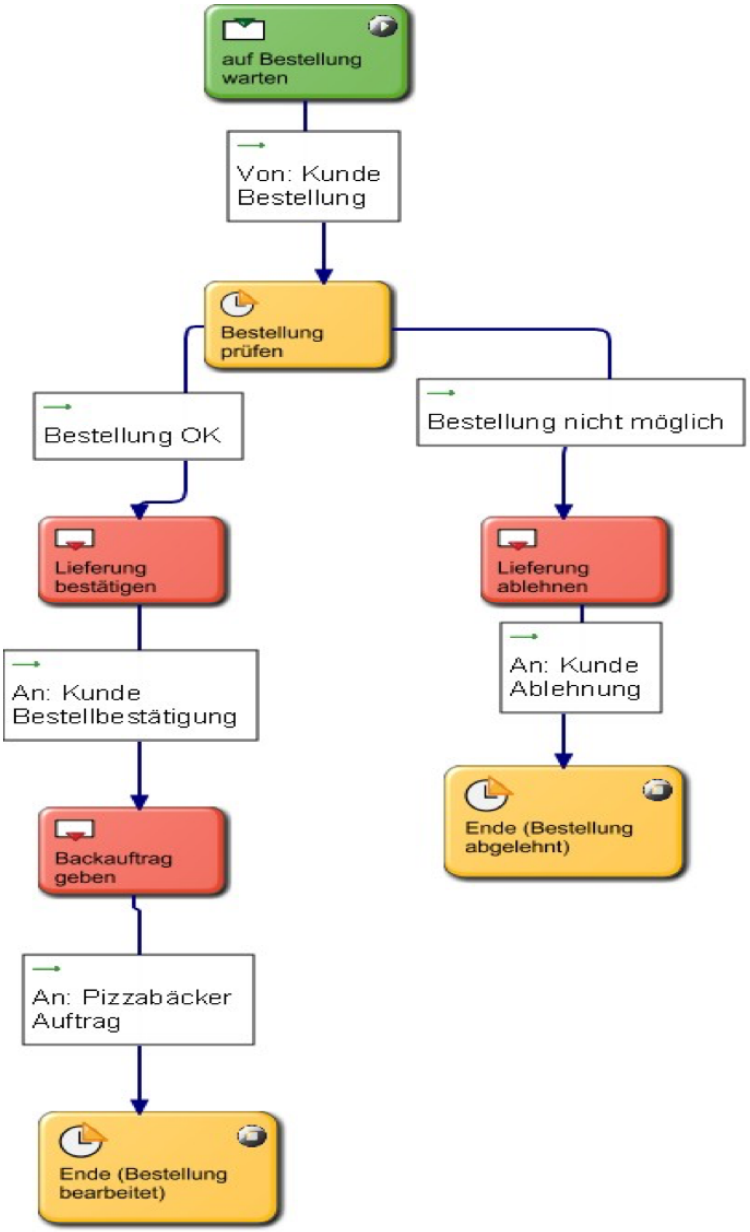


Abbildung 2.4: Interne Ansicht des Telefonisten

Der Telefonist wartet zu Beginn des Prozesses auf den Bestelleingang des Kunden. Wenn die Bestellung eingegangen ist, sendet der Telefonist entweder eine Ablehnung, falls die Bestellung nicht möglich sein sollte und beendet damit den internen Prozess, oder der Kunde erhält eine Bestellbestätigung. Nachdem die Bestätigung erteilt wurde, wird der Auftrag an den Pizzabäcker weitergeleitet und der interne Prozess ist auch hier beendet.

Als nächstes wird der interne Prozess des Pizzabäckers betrachtet:

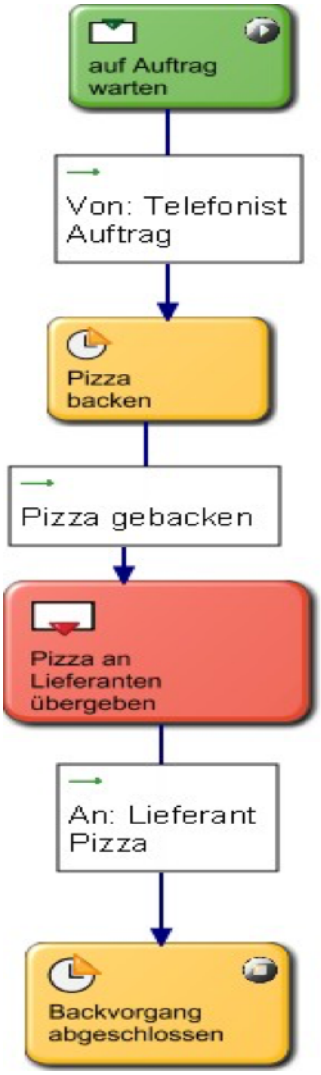


Abbildung 2.5: Interne Ansicht des Pizzabäckers

Im Vergleich zu den anderen Prozessen kommt der Verlauf des Pizzabäckers ohne Verzweigungen aus. Der Pizzabäcker wartet zu Beginn auf den Auftragseingang, nachdem dieser vom Telefonisten erteilt wird, wird die Pizza gebacken und im Anschluss an den Lieferanten übergeben. Damit ist dann auch bereits der Prozess beendet.

Zum Schluss wird das internen Verhalten des Lieferanten modelliert.

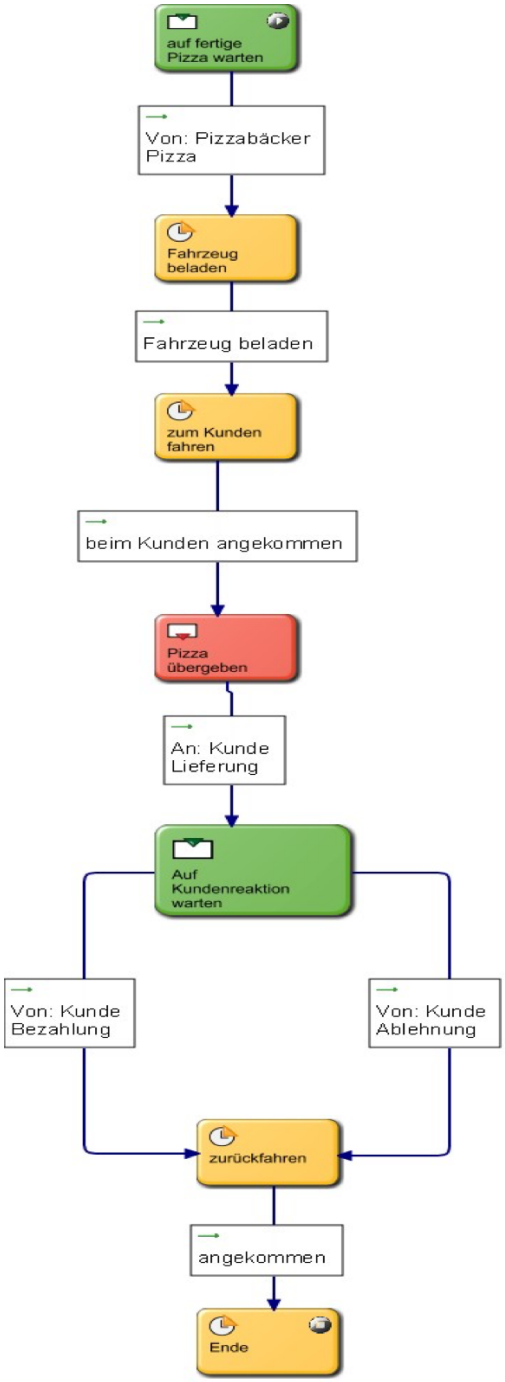


Abbildung 2.6: Interne Ansicht des Lieferanten

Der Lieferant wartet auf die fertige Pizza des Pizzabäckers, lädt diese in sein Auslieferungsfahrzeug und fährt damit zum Kunden. Wenn der Lieferant beim Kunden angekommen ist, wird dem Kunden die Pizza übergeben. Wie bereits im Prozess des Kunden beschrieben, trifft dieser dann die Entscheidung, ob die Pizza in Ordnung ist und der Lieferant erhält eine Bezahlung oder eine Ablehnung vom Kunden. In beiden Fällen fährt der Lieferant danach zurück zur Pizzeria und beendet den Gesamtprozess.

2.3.5 Simulation der Prozeesse mit S-BPM

2.3.6 Potenzielle Fehler

Die Pizzabestellung kann Fehler im Prozess aufweisen. Mögliche Extrembeispiele wie einen Telefonstreich, der Empfänger ist zu weit entfernt, das Auslieferungsfahrzeug hat einen Platten, die Lieferadresse ist falsch usw. verdeutlichen, dass der Prozess, welcher eventuell auf den ersten Blick einfach wirkt, sehr schnell kompliziert und groß werden kann. Nun muss man sich überlegen, welche fehlertoleranten Prozesse notwendig sind, damit der Prozess einwandfrei laufen kann. Das Modell für die Pizzabestellung und die erwähnten möglichen Fehler im Prozess sollen zeigen, dass eine einfache Modellierung nicht ausreicht, um einen Prozess einwandfrei darzustellen. Im nächsten Abschnitt wird auf die Fehlertoleranzproblematik genauer eingegangen.

2.4 Fehlertoleranzproblematik

Ein wichtiges Ziel beim Entwurf von Prozessen in verteilten Systemen ist, dass diese bei partiellen Ausfällen weiterarbeiten können und die Gesamtleistung dabei nicht bemerkbar beeinträchtigt wird. Wenn ein Fehler auftritt, sollte das verteilte System während der Durchführung von Reparaturen weiterarbeiten können. Es sollte also Fehler tolerieren und zu einem gewissen Grad weiterarbeiten.
In diesem Kapitel betrachten wir grundlegende Aspekte über die Fehlertoleranz. Diese wären die Klassifizierung der Fehler, die Realisierung der Fehlertoleranz in verteilten Systemen, die Organisation der Prozesse in Gruppe und zuletzt die Übereinstimmung in fehlerhaften Systemen.

2.4.1 Verlässliche Systeme

Fehlertoleranz hängt eng mit verlässlichen Systemen zusammen. Der Begriff Verlässlichkeit deckt unterschiedliche Anforderungen an verteilte Systemen ab, unter anderem die

folgenden: Verfügbarkeit (Availability), Zuverlässigkeit (Reliablity), Funktionssicherheit (Safety), Wartbarkeit (Maintainability).
Die *Verfügbarkeit* hat die Eigenschaft, dass ein System sofort benutzbar ist. Mit anderen Worten ein höchstverfügbares System funktioniert zu jedem beliebigen Zeitpunkt.
Die *Zuverlässigkeit* beschreibt, dass ein System forlaufend ausfallfrei läuft. Ein höchst zuverlässiges System arbeitet durchgehend ohne Unterbrechung über eine lange Zeitspanne. Die Zuverlässigkeit ist für ein Zeitintervall definiert und nicht wie bei der Verfügbarkeit für einen bestimmten Zeitpunkt. Wenn ein Server also jede Stunde eine Millisekunde ausfällt dann ist die Verfügbarkeit hoch, ist aber höchst unzuverlässig. Ein Server, der nie abstürzt, aber in jedem Jahr im Dezember heruntergefahren wird, ist höchst zuverlässig und dafür weniger verfügbar.
Die *Funktionssicherheit* beschreibt den Umstand, dass die Funktion aufrecht gehalten werden kann, auch wenn Komponenten ausfallen.
Die letzte Anforderung ist die *Wartbarkeit*, diese sagt aus, wie leicht und schnell ein ausgefallenes System repariert werden kann. Ein höchst wartbares System kann eine hohe Verfügbarkeit aufweisen, vor allem, wenn Ausfälle automatisch erkannt und repariert werden können. Tiefergehende Informationen erhält man aus der einführenden Informatikliteratur zu verteilten Systemen (zum Beispiel von der Universität Kiel und Technischen Universität München), aus denen dieser Zugang entnommen wurde.

2.4.2 Fehlermodelle

Ein System stellt die Dienste bei einem Ausfall nicht mehr angemessen zur Verfügung. Wenn man ein verteiltes Sytem als Ansammlung von Servern betrachtet, die miteinander kommunizieren, dann bedeutet eine nicht angemessene Bereitstellung der Dienste, dass Server, Kommunikationskanäle oder beides nicht funktionieren. Es wurden Fehlerklassen für ein besseres Verständnis entwickelt und um Fehlertoleranzen von Programmen darauf zu beziehen. Eine solche Klassifikation wird in der Tabelle 2.1 basierend auf den in Cristian (1991) und Hadzilacos und Toueg (1993) (Hadzilacos and Toueg [1993]) beschriebenen Verfahren veranschaulicht.

Ausfallart	Beschreibung
Absturzausfall (Crash Failure)	Ein Server steht, hat aber bis dahin korrekt gearbeitet. Der angebotene Dienst bleibt beständig aus (ständiger Dienstausfall).
Dienstausfall (Omission Failure) Empfangsauslassung (Receive omission) Sendeauslassung (Send omission)	Ein Server antwortet nicht auf eingehende Anforderungen. Ein Server erhält keine eingehenden Anforderungen. Ein Server sendet keine Nachrichten.
Zeitbedingter Ausfall (Timing Failure)	Die Antwortzeit eines Servers liegt außerhalb des festgelegten Zeitintervalls.
Ausfall korrekter Antwort (Response Failure) Ausfall durch Wertfehler (Value Failure) Ausfall durch Zustandsübergangsfehler (State Transition Failure)	Die Antwort eines Servers ist falsch. Der Wert der Antwort ist falsch. Der Server weicht vom richtigen Programmablauf ab.
Byzantinischer oder zufälliger Ausfall (Byzantine oder Arbitrary Failure)	Ein Server erstellt zufällige Antworten zu zufälligen Zeiten.

Tabelle 2.1: Unterschiedliche Fehlerarten

Ein *Absturzausfall* (Crash Failure) tritt auf, wenn ein Server ausfällt, aber bis zum Absturz richtig gearbeitet hat. Ein wichtiger Aspekt des Absturzausfalls ist, das sich der Server nach dem Absturz nicht mehr meldet.
Ein *Dienstausfall* (Ommision Failure) tritt auf, wenn ein Server auf eine Anforderung nicht antwortet. Bei einem *Empfangs-Auslassungsfehler* wird der aktuelle Zustand des Servers nicht beeinflusst, weil der Server nicht wahrnimmt, dass eine Nachricht gesendet wurde. Bei einem *Sende-Auslassungsfehler* hat der Server seine Arbeit geleistet, aber kann keine Antwort senden. Hierbei muss man beachten, wenn der Server keine Antwort senden kann, dass der Server eventuell eine wiederholte Anforderungen empfangen wird. Eine weitere Ausfallfehlerklasse bezieht sich auf zeitliche Rahmenbedingungen (Timing Failure). *Zeitbedingte Ausfälle* treten auf, wenn die Antwort außerhalb eines vorgegebenen Zeitintervalls erfolgt. Zum Beispiel kann eine frühe Bereitstellung der Daten für einen Empfänger leicht problematisch werden, wenn nicht ausreichend Pufferspeicher zur Verfügung steht, um alle eingehenden Daten aufzunehmen. Eine wichtige Fehlart ist ein *Ausfall korrekter Antwort* (Response Failure), bei dem die Antwort eines Servers falsch ist. Es können zwei Arten von Ausfällen auftreten. Ein *Ausfall durch einen Wertfehler* (Value Failure) kann beispielsweise bei einer Suchanfrage vorkommen, bei der das Ergebnis sich nicht auf die verwendeten Suchbegriffe bezieht. Die andere Art von Ausfall korrekter Antwort kann durch einen *Zustandsübertragungsfehler* (State Transition Failure) ausgelöst werden. Diese Art von Ausfall tritt auf, wenn ein Server beispielsweise eine Nachricht empfängt, die er nicht erkennen kann und dadurch keine Maßnahmen getroffen

werden.
Bei einem *byzantinischen Ausfall* (Byzantine Failure) kann es vorkommen, dass ein Server Ausgaben generiert, die er nie erstellen sollte und diese nicht als fehlerhaft erkennt. Schlimmstenfalls kann ein Server vorsätzlich anderen Servern falsche Antworten senden.

2.4.3 Maskierung von Fehlern mittels Redundanz

Die beste Vorgehensweise ein System fehlertolerant zu entwickeln ist, das Auftreten von Ausfällen vor anderen Prozessen zu verbergen (Maskierung). Die Maskierung der Fehler wird durch die Verwendung von Redundanz erreicht. Drei Arten sind möglich: Informationsredundanz, zeitliche Redundanz und technische Redundanz (siehe auch Johnson [1996]).
Bei der *Informationsredundanz* werden zusätzliche Bits verwendet, um den möglichen Ausfall anderer Bits abzufangen. Beispielsweise kann ein Hamming-Code zu den übertragenen Daten hinzugefügt werden, um Fehler durch Rauschen auf der Übertragungsleitung wiederherstellen zu können.
Bei der *zeitlichen Redundanz* wird eine Aktion bei Erforderlichkeit nach der Ausführung wiederholt. Die zeitliche Redundanz ist bei vorübergehenden und wiederkehrenden Fehlern sehr hilfreich.
Bei der *technischen Redundanz* werden zusätzliche Hardware oder Prozesse verwendet, damit das System als Ganzes den Verlust oder die Fehlfunktion bestimmter Komponenten ersetzen kann. Beispielweise können zu dem System weitere Prozesse hinzugefügt werden, sodass es auch bei einer kleinen Anzahl von Abstürzen weiterhin korrekt arbeiten kann. Durch die Replikation von Prozessen ist ein hohes Maß an Fehlertoleranz erreichbar. Die technische Redundanz kann man mit verschiedenen Aspekten aus der Umwelt assoziieren, zum Beispiel in der Biologie (neben dem Raumaspekt, zwei Augen, zwei Ohren usw.) oder der Einsatz von drei Schiedsrichtern beim Fußball.

2.4.4 Gruppenorganisation

Um einen fehlerhaften Prozess tolerieren zu können, ist eine Anordnung identischer Prozesse in einer Gruppe ein wichtiger Ansatz. Zweck des Einsatzes von Gruppen ist eine Darstellung als eine Einheit gegenüber der Außenwelt, also empfangen alle Mitglieder einer Gruppe eine Nachricht, die an die Gruppe selbst gerichtet ist. Wenn ein Prozess in einer Gruppe ausfällt, können andere Prozesse für ihn einspringen (Guerraoui and Schiper [1997]).
Ein weiterer Zweck der Einführung von Gruppen besteht darin, eine Reihe von Prozessen

als eine Abstraktion zu betrachten. Das heißt ein Prozess kann eine Nachricht an eine Gruppe von Prozessen senden, ohne zu wissen, wie viele es gibt oder wo sie sich befinden (Multicast).

Die Gruppenorganisation kann sich aus einer *strukturlosen Gruppe* oder *hierarchischen* zusammenbilden. In der strukturlosen Gruppe sind alle Prozesse gleich. Keiner der Prozesse ist über- oder untergeordnet und alle Entscheidungen werden gemeinsam getroffen. In der hierarchischen Gruppe gibt es einen Prozess der *Koordinator* ist und alle anderen sind die *Arbeiter*. Wenn in diesem Modell entweder ein externer Client oder einer der Arbeiter eine Arbeitsanforderung stellt, wird sie an den Koordinator weitergegeben. Der Koordinator gibt die Anforderung dorthin weiter, der für ihn am besten für die Ausführung geeignet ist.

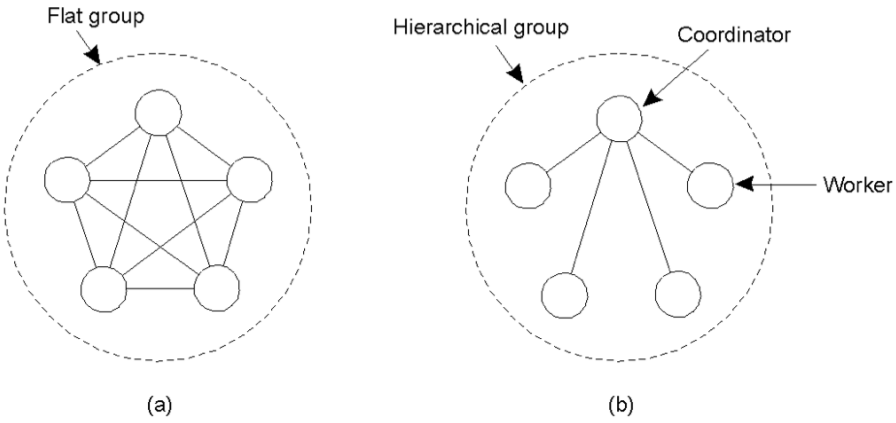


Abbildung 2.7: (a) Kommunikation in einer strukturlosen Gruppe. (b) Kommunikation in einer hierarchischen Gruppe. (aus Tanenbaum and van Steen [2007])

Beide Gruppenorganisationen haben verschiedene Vor- und Nachteile. Die strukturlose Gruppe ist symmetrisch und hat keinen einzelnen Ausfallpunkt. Falls einer der Prozesse ausfällt, wird die Gruppe kleiner, aber kann so weiterarbeiten. Ein Nachteil wäre, dass jede Entscheidung eine Abstimmung erfordert, was zu Verzögerung und extra Aufwand führt. Die hierarchische Gruppe hat dagegen - durch die Entscheidung von nur einem Prozess - einen schnellen Normalbetrieb. Ein Ausfall dieses Prozesses kann jedoch die gesamte Gruppe zum Stehen bringen.

2.4.5 Übereinstimmung in fehlerhaften Systemen

Nun ist es wichtig zu betrachten, wie es bei einer durchzuführenden Aktion im Fall *fehlerhafter Prozesse* (aber perfekter Kommunikation) zu einer Übereinstimmung kommt. Dieses Problem und die dazugehörige Lösung lässt sich mittels des *Problems der byzantinischen Generäle* in analoger Weise darstellen: Eine rote Armee im Tal soll angegriffen werden. In den Hügeln sind n blaue Armeeteile. Die Kommunikation ist über eine zuverlässige Verbindung zwischen allen n Generälen hergestellt. Problematisch ist, dass k der n Generäle Verräter sind und versuchen, die Übereinstimmung der anderen zu verhindern. Ziel der byzantinischen Übereinstimmung besteht darin, einen Konsens über die Werte der nicht fehlerhaften Prozesse zu erreichen. Lamport et al. (1982) hat bewiesen, dass in einem System mit k fehlerhaften Prozessen nur dann eine Übereinstimmmung erzielt werden kann, wenn es insgesamt $3k+1$ Prozesse gibt. Also benötigen wir bei einem einzigen fehlerhaften Prozess drei weitere richtig funktionierende Prozesse, damit eine Übereinstimmung gefunden werden kann ($2k+1$ korrekte Aussagen von $3k+1$). Mit steigender Anzahl von k ist eine Übereinstimmung nur dann möglich, wenn zwei Drittel der Prozesse richtig arbeiten. In dem folgenden Beispiel geht es darum, eine Übereinstimmung der Truppenzahl zu erreichen.

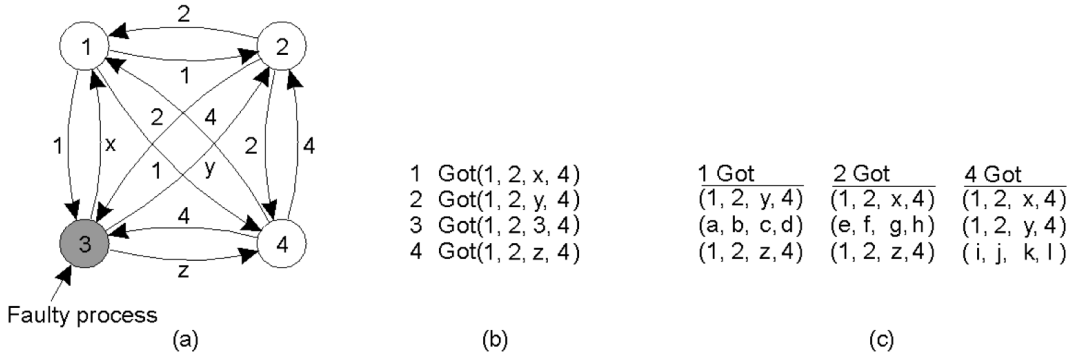


Abbildung 2.8: Das byzantinische Übereinstimmungsproblem für drei funktionierende Prozesse und einen fehlerhaften Prozess. (a) Jeder sendet einen Wert an die anderen. (b) Ergebnis in Vektor. (c) Vektoren, die jeder Prozess aus Schritt 3 empfängt. (aus Tanenbaum and van Steen [2007])

Szenario mit n=4 und k=1. **Annahme**: G1, G2 und G4 nennen die korrekte Stärke, G3 belügt alle.
Mit diesen Parametern werden 4 Schritte gebraucht, um das Problem zu lösen.
Schritt 1: Alle Generäle melden ihre Stärke an alle anderen. In Abb. 2.8 (a) sehen wir, dass Prozess 1 den Wert 1 meldet, Prozess 2 den Wert 2, Prozess 3 belügt alle und liefert x, y bzw. z und Prozess 4 meldet einen Wert von 4.

Schritt 2: Die Ergebnisse der Ankündigungen von Schritt 1 werden in Form des in Abb. 2.8 (b) dargestellten Vektors erfasst.

Schritt 3: Jeder General meldet seinen Vektor aus Abb. 2.8 (b) an jeden anderen General. Hier lügt wieder General 3 und führt zwölf neue Werte a bis ℓ ein. Die Ergebnisse von Schritt 3 werden in Abb. 2.8 (c) gezeigt.
Schritt 4: Jeder General überprüft die Spalten: Wenn es eine Mehrheit gibt, ist der Wert korrekt und wird in den Ergebnisvektor gestellt. Falls kein Wert eine Mehrheit hat, wird das entsprechende Element des Ergebnisvektors als Unbekannt gekennzeichnet.
Übereinstimmung: (1, 2, Unbekannt, 4)

2.4.6 Fehlerdetektion

Mit der Modell-Suite von Metasonic kann eine Simulation der S-BPM-Prozesse durchgeführt werden, um die einzelnen Prozessschritte zu durchlaufen. Damit ist der Prozessablauf mit allen seinen Verzweigungen abarbeitbar. Deadlocks können so aufgespürt werden.
Es stellt sich weiterhin die Frage nach der Beweisbarkeit. Diese kann in der Arbeit jedoch nicht beantwortet werden, da der Schwerpunkt im Bereich der grundlegenden Modellierung liegt.

3 Flooding

Fabian Sell

3.1 Einleitung

In diesem Abschnitt des Projekts soll das Thema Flooding näher beleuchtet werden. Unter Flooding versteht man in der Netzwerkkommunikation die Nachrichtenweitergabe an alle direkt verbundenen Netzwerkknoten. Dabei dienen angrenzende Netzwerkknoten als Relay, um die Nachricht im Netzwerk zu verteilen. Das bedingt, wie im Abschnitt 2.2 zum Thema MANET bereits erläutert, dass jeder Netzwerkknoten alle Funktionalitäten haben muss, um sowohl Nachrichten zu versenden, als auch zu empfangen.
In der simpelsten Form werden die Nachrichten ohne eine Empfängeradresse an alle Knoten versendet. Es ist dann erweitert möglich eine Empfängeradresse anzugeben, sollte die Nachricht einen bekannten Empfänger haben. In einer weiteren Form können bei dem Versand von Nachrichten zusätzlich jeweils die eigene Netzwerkadresse oder Netzwerkkennung übermittelt werden und im nächsten Schritt zusätzlich noch die Last Hop Address, also die Netzwerkadresse des Knotens, von dem der Versender die Nachricht erhalten hat. Jeder Knoten prüft die eingegangenen Nachrichten und sendet diese wieder aus. Wurde er adressiert, dann wird zusätzlich die eingegangene Nachricht von ihm interpretiert. Dies ermöglicht auch dem Relay-Knoten, Nachbarschaftskenntnisse aufzubauen. Bei dem Restricted Flooding werden auch sowohl Versender als auch Last Hop Address versendet. Der Versender kennt sein Netzwerk dahingehend, dass er entscheiden kann, über welchen Weg die Nachricht zum Empfänger und entsprechend seine Empängeranzahl einschränken kann und zum Beispiel nur in einen Korridor flutet (GUWMANET-Ansatz).

3.2 Erste Prozessformulierung

Nach der ersten Einarbeitung in das Thema Flooding haben wir im Team überlegt, welche Funktionalitäten unser Netzwerk erfüllen können muss. Da unter Wasser nur eine sehr begrenzte Anzahl an Bits für die Übertragung zur Verfügung stehen, mussten wir unsere Anforderungen sehr minimalistisch halten. Die einfachste Form nach dem Prinzip *fire*

and forget kann durch Logik im Kommunikationsknoten verbessert werden. Auch ohne dedizierte Adressierung in den Nachrichten kann die Weiterleitung der eigenen Aussendung in der Ein-Hop-Nachbarschaft als Empfangsnachricht wiedererkannt werden. Diese From wird als *implizites Acknowledgement* bezeichnet. Damit wird bestätigt, dass die ausgestrahlte Nachricht bei der Ein-Hop-Nachbarschaft korrekt eingetroffen ist. Auch kann mit der Logik des Vergleichs von aktueller und historischer Nachrichten festgestellt werden, dass die vorliegende Nachricht bereits ausgestrahlt wurde (Gedächtnis). Eine Endlosversendung ist damit ausgeschlossen. Wird zwangsläufig eine Antwort erwartet, wird von einem *expliziten Acknowledgement* gesprochen. Mit dem Erhalt der eigenen Nachricht durch die Weiterleitung der unmittelbaren Nachbarn als *implizites Acknowledgement* wird sichergestellt, dass die Nachricht mindestens von einem anderen Knoten erhalten wurde.
Zusätzlich sollen grundsätzlich alle Nachrichten dreimal versendet werden, um eine grundlegende *Redundanz* zu bieten. Auch wenn das keine häufige Wiederholung darstellt, zum Vergleich, Nachrichten im Internet öfters versendet, wollen wir damit eine verbesserte Nachrichtenzustellung erreichen. Eine dreimal versendete Nachricht wird selbständig nicht erneut versendet, auch wenn diese später erneut eintrifft (Gedächtnis). Im Weiteren werden Prüfwerte wie Checksummen respektive CRCs (cyclic redundancy check - zyklische Redundanzprüefung) benötigt. Stellvertretend wird von Checksummen gesprochen. *Checksummen* werden eingeführt, um eine Integrität der Nachrichten zu gewährleisten. Die Checksummen sollen außerdem in einer Tabelle gespeichert werden, damit anhand der Checksumme Nachrichten mit hoher Wahrscheinlichkeit zuzuordnen werden. Aufgrund der kurzen Längen ist diese Beziehung nicht eindeutig. Insgesamt soll das Flooding die Schnittstelle zwischen den Applikationen und den anderen Knoten im Netzwerk sein und somit eine Kommunikation zwischen *Autonomous Underwater Vehicles (AUVs)*, *Glidern*, *Boden- und Sensorknoten* und *Gateways* ermöglichen.
Um dem Leser nun das Prinzip der Next Hop Nachbarn zu verdeutlichen, wurde nachfolgende Grafik erstellt:

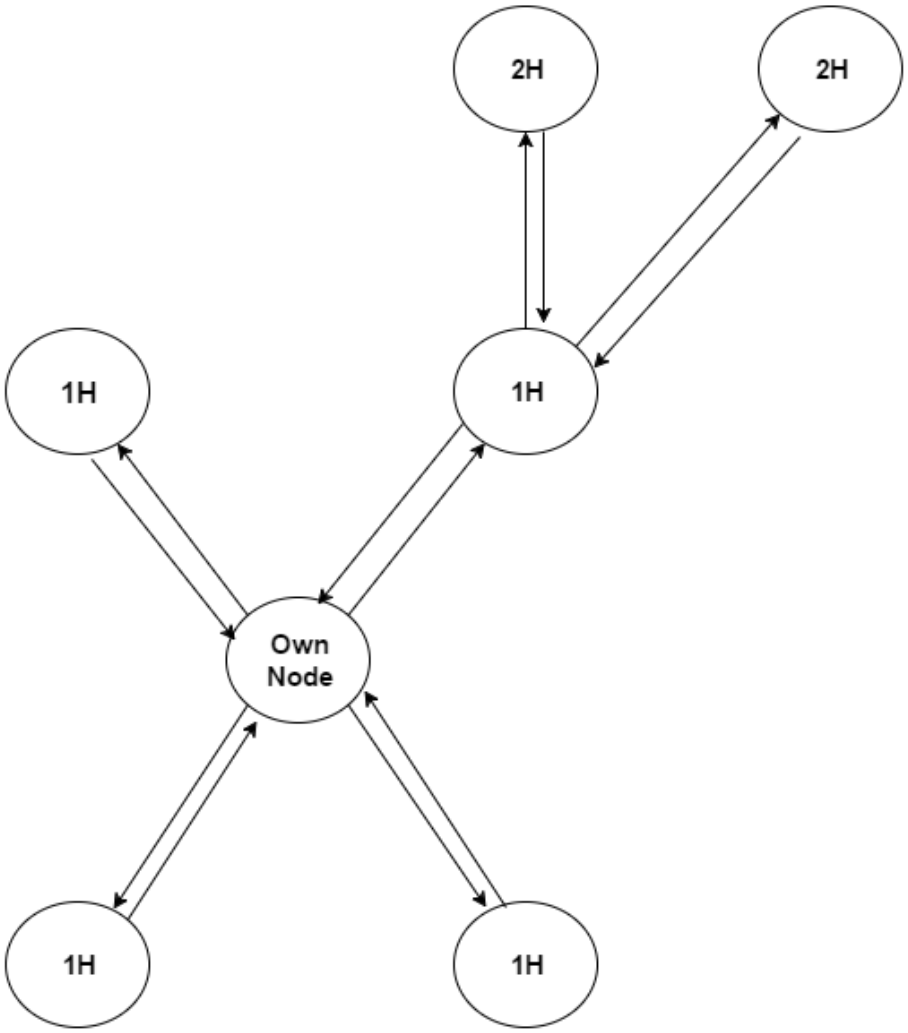


Abbildung 3.1: Next Hop Nachbarn

Der Own Node bezeichnet in diesem Fall den Absender der Ursprungsnachricht, alle mit **1H** gekennzeichneten Knoten sind Einhop-Nachbarn, von denen der Absender ein implizites Acknowledgement bekommt. 1H oben rechts sendet außerdem die Nachricht noch an die beiden **2H** Knoten, also sogenannte Zweihop-Nachbarn des Own Nodes, von diesen beiden Knoten erhält der Own Node kein Acknowledgement, sondern nur der Knoten 1H, von dem die Nachricht versandt wurde.

3.3 Komplikationen beim Flooding

Ein großes Problem der Unterwasserkommunikation ist, dass es keine internationalen Standards für ein Netzwerkprotokoll gibt. Auf europäischer Seite wird unter dem Namen RACUN (Robust Acoustic Communication in Underwater Networks) in einem Konsortium an einem möglichen Standard gearbeitet, bis jetzt wurde dieser allerdings noch nicht festgelegt. Man konnte allerdings bereits mehrere Kommunikationsprotokolle entwickeln und auch schon unter Realbedingungen testen.
Die Naval Postgraduate School forscht in den USA an einem Unterwassernetzwerksystem, das Seaweb genannt wird. Ursprünglich dafür gedacht, U-Boote in Küstennähe aufspüren zu können, wird inzwischen in weitere Richtungen geforscht, in vielen Fällen aber militärisch.
International gibt es ein ISO Komitee, das sich unter der Norm ISO/IEC CD 30140-X mit der Entwicklung eines Standards für die Unterwasserkommunikation von Sensoren beschäftigt. Hier gibt es allerdings noch keine veröffentlichten Ergebnisse.
Insgesamt ist also gar nicht klar, wie die Netzwerkkommunikation genau funktionieren kann und soll, da erschwerend hinzukommt, dass das Wasser als Übertragungsmedium zahlreiche Übertragungsfehler begünstigt. Man merkt insgesamt, dass die Forschungsarbeiten rund um die Unterwasserkommunikation relativ jung sind und erst um die Jahrtausendwende gestartet sind.
Nachfolgend sollen ein paar allgemeine Komplikationen in der Unterwasserkommunikation erläutert werden, auf die dann weiter im Abschnitt 3.6 auf Seite 45 eingegangen werden soll.

Verteilte Systeme: Bei den Netzwerkknoten handelt es sich um verteilte Systeme, das heißt, dass die Netzwerkknoten ihre Nachbarknoten nicht zwangsläufig kennen. Als Vergleich lässt sich hier das Zwei-Armeen-Prinzip anführen (siehe auch Abschnitt 2.4.5), bei dem ein Teil des Systems nicht wissen kann, was der andere Teil gerade tut und nicht entsprechend reagieren kann. Das erschwert die Direktion von Nachrichten entsprechend weiter, gerade dadurch, dass die Kommunikation möglichst minimalistisch gehalten werden soll. Auch aus diesem Grund wurde das Restricted Flooding außen vor gelassen.

Übertragungszeit: Ein weiteres Problem ist, dass die Schallgeschwindigkeit unter Wasser nicht linear verläuft und zum Beispiel von dem Salzgehalt, der Temperatur und dem

Druck des Wassers beeinflusst wird.Im Vergleich zur Lichtgeschwindigkeit, ist der Schall auch deutlich langsamer, sodass Verzögerungen schnell in den Sekundenbereich gehen. Aus Gründen der Vereinfachung nehmen wir in den nachfolgenden Kapiteln eine Schallgeschwindigkeit von 1500 Metern pro Sekunde an.

Identifizierung: Im Vergleich zu zum Beispiel Internetprotokollen ist in der Unterwasserkommunikation keine Kapazität für eine detaillierte Adressierung in den Nachrichten vorhanden, daher muss auf eine andere Möglichkeit der Nachrichtenerkennung zurückgegriffen werden. Es könnte sonst zu dem Fall kommen, dass man seine eigenen Nachrichten beim Relaying dauerhaft versendet.

Netzwerküberlastung: Es ist in der Theorie denkbar, dass das Netzwerk durch zu viele versendete Nachrichten überschwemmt wird und dadurch das Netzwerk zum Erliegen gebracht werden könnte.

Nachrichten nicht erhalten oder nicht vollständig: Aus den oben angeführten Gründen, kann es passieren, dass eine Nachricht nicht oder nur unvollständig erhalten werden kann. Auch mit diesem Fall muss gerechnet werden.

Fakenews: Es besteht die Möglichkeit, dass Dritte mit den Knotenpunkten kommunizieren und falsche Informationen versenden.

3.4 Nachrichten im Flooding

Da zwar keine fertigen Standards für die Unterwasserkommunikation existieren, es aber sehr wohl Anwendertelegramme gibt, sollen diese im Folgenden kurz vorgestellt werden, da in den nachfolgenden Kapiteln vertiefend auf diese Nachrichtentypen eingegangen wird.

JANUS

Das Janus-Projekt [1] ist ein Open Source Projekt, das sich unter der Schirmherrschaft des Centre for Maritime Research and Experimentation in Italien um die Entwicklung einer Unterwasserkommunikationsmöglichkeit bemüht, die möglichst robust ist. Das Janus-Protokoll sieht eine Nachrichtengröße von nur 64 Bit vor, zum Vergleich mit dem alltäglichen Leben stehen einer SMS 1.120 Bit zur Verfügung. Daher eignet sich das Janus-Protokoll vor allem zu der Erstkontaktaufnahme. Dies soll im nächsten Kapitel ausgeführt werden. Dadurch, dass es sich um ein Open Source Projekt handelt, können die Funktionalitäten laufend erweitert werden. Bis jetzt wurde das Nachrichtenprotokoll weltweit getestet und bietet eine Reichweite von bis zu 28 Kilometern. Um eine Vorstellung von der Bit-Zuordnung in einer Janus-Nachricht zu erhalten, ist nachfolgend die Janus Bit Allocation Table (Potter et al. [2014]).

JANUS BIT ALLOCATION TABLE

Bits	Descriptor	Values	Comments
1-4	Version	0011	unsigned 4 bit integer. Current version is 3
5	Mobility flag	0=static, 1=mobile	Indicates nature of the transmitting platform
6	Schedule flag	0=off, 1=on	If On (set to 1), the first bit in the Application Data Block (ADB) indicates if the interval is to be interpreted as a reservation time (bit set to 0) or a repeat interval (bit set to 1). The time is specified from (different) look-up tables in bits 2-8 of the ADB, as specified in Annexes B and C
7	Tx/Rx Flag	0=Tx-only, 1=Tx/Rx	Tx-only implies at least the ability to detect energy in band to satisfy the MAC requirements. Tx/Rx implies not only detect, but decode capability.
8	Forward capability	0=no, 1=yes	Used for routing and Delay Tolerant Networking
9-16	Class user i.d.	[00000000 : 11111111]	Allows 256 classes of users, mostly individual nations
17-22	Application Type	[000000 : 111111]	Allows 64 different types of message per user i.d. class user specified
23-56	Application Data Block	Determined by user	if the schedule flag (bit 6) is set, the first 8 bits are dedicated to defining the nature of the schedule (reserved or repeat interval) and time in seconds from a lookup table.
57-64	8-bit Checksum		8-bit CRC run on the previous 56 bits with polynomial $p(x) = x^8 + x^2 + x^1 + 1$, init=0

Abbildung 3.2: Bit Allocation Table der Janus Nachrichten

Der Nachricht ist also zum Beispiel zu entnehmen, ob der Versender sich unter Wasser bewegen kann, oder ein fest installierter Knoten ist. Man sieht ebenfalls, wie wenig Bit dem Benutzer für den Dateninhalt zur Verfügung stehen. Bei Janus sind dies nur 34 Bit. Weitere acht Bit stehen für die Generierung einer CRC-Prüfzahl zur Verfügung.

GUWAL

Außerdem gibt es noch GUWAL-Nachrichten[2], die dem Versender 128 Bit Nachrichtenlänge zur Verfügung stellen. Nachfolgend dazu ebenfalls die Bit-Zuordnung anhand einer

[1] siehe auch `http://www.januswiki.com`
[2] GUWAL steht für Generic Underwater Application Language

Tabelle und anschließend eine beispielhafte Nachricht um einem AUV GPS-Koordinaten zu senden (Nissen and Goetz [2012]).

Position	Length	Field
1-2	2	Parcel Type
3	1	End-to-End Acknowledgment
4	1	Priority Flag
5-10	6	Operational Source Address
11-16	6	Operational Destination Address
17-112	96	Payload
113-128	16	Checksum

Abbildung 3.3: Allgemeiner Aufbau einer GUWAL Nachricht

Im Vergleich zu den Janus-Nachrichten bietet GUWAL fast die dreifach Menge an Bit für den Nachrichten-Payload, sowie die doppelte Anzahl an Bit um eine CRC-Prüfzahl zu generieren. Des Weiteren sieht GUWAL Absender- und Zieladressierung auf Applikationsebene vor.

Position	Bits	Verwendung
0-1	2	Kontrollnachricht
2	1	End-zu-End Bestätigungs-Flag (Acknowledgement flag)
3	1	Prioritätsmarkierung
4-9	6	betriebliche Startadresse
10-15	6	betriebliche Zieladresse
16-34	19	Zeitstempel des Kommandos
35-39	5	Kommandotyp
40-42	3	**Fortbewegungsart**
43-66	24	**Breite der GPS-Zielposition (y)**
67-90	24	**Länge der GPS-Zielposition (x)**
91-95	5	**Zieltiefe (t)**
96-107	12	**Ankunftszeit** am Endpunkt
108-110	3	**Startpunkt**
111	1	**Flag: Position zu Route hinzufügen?**
112-127	16	Checksumme

Tabelle 3.1: „move-to“-Nachricht

In diesem Beitrag wird eine „Here I am“ oder auch „move-to“-Nachricht genutzt. Diese werden später in den Kapiteln 5 (Distance Measurement) und 7 (Task Handover) verwendet, um anderen AUVs ihre Zielposition mitzuteilen, oder den zu scannenden Bereich zu übermitteln.

3.5 Modellierung in S-BPM

In diesem Abschnitt sollen die modellierten Prozesse vorgestellt werden. Dadurch, dass wir hier nur die Netzwerkebene betrachten, die wie angesprochen sehr minimalistisch gehalten ist, sind die Prozesse an dieser Stelle überschaubar. Wir haben für das Projekt zwei Szenarien modelliert, zum einen die fire and forget Methode des Erstversands einer neuen Nachricht, und zum anderen das Prinzip des Relays über mehrere Hops.

3.5.1 Fire and forget

Zuerst soll die „fire and forget“-Methode beleuchtet werden, beginnend mit der Kommunikationssicht und anschließend den internen Ansichten von Sender und Empfänger, da dies die Grundlage unseres Floodings darstellt.

Kommunikationssicht

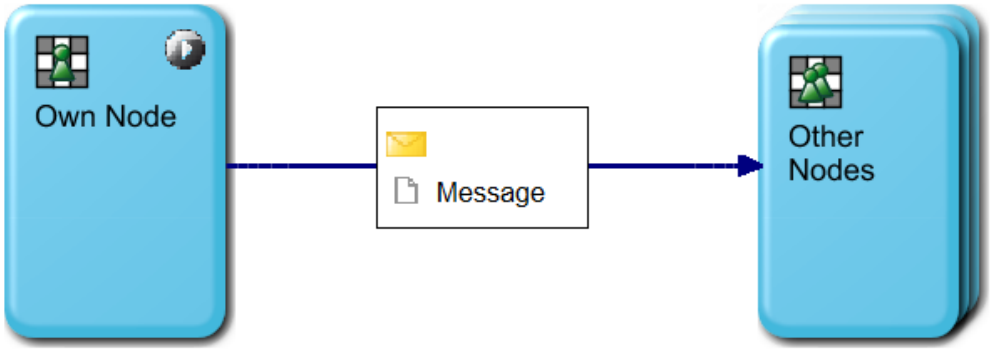


Abbildung 3.4: Kommunikationssicht des Floodings

Die Kommunikationssicht des „fire and forget“-Prinzips ist äußerst simpel aufgebaut.

Es gibt nur zwei Subjekte in dem Prozess, wobei das Subjekt der „Other Nodes “ ein sogenanntes Multisubjekt ist, also eine Vielzahl von Objekten, die hier stellvertretend für alle Nachbarknoten stehen. Wie in Abbildung 3.4 zu sehen ist, gehen die Nachrichten von dem Own Node an die anderen Knoten aus. Es gibt in der Metasonic Suite leider keine gelungene Möglichkeit das implizite Acknowledgement für das Multisubjekt zu modellieren, die Suite sieht auch nicht vor, dass einzelne Knoten nicht antworten könnten. Daher muss diese in den internen Ansichten textuell beschrieben werden.

Interne Ansicht des Own Nodes

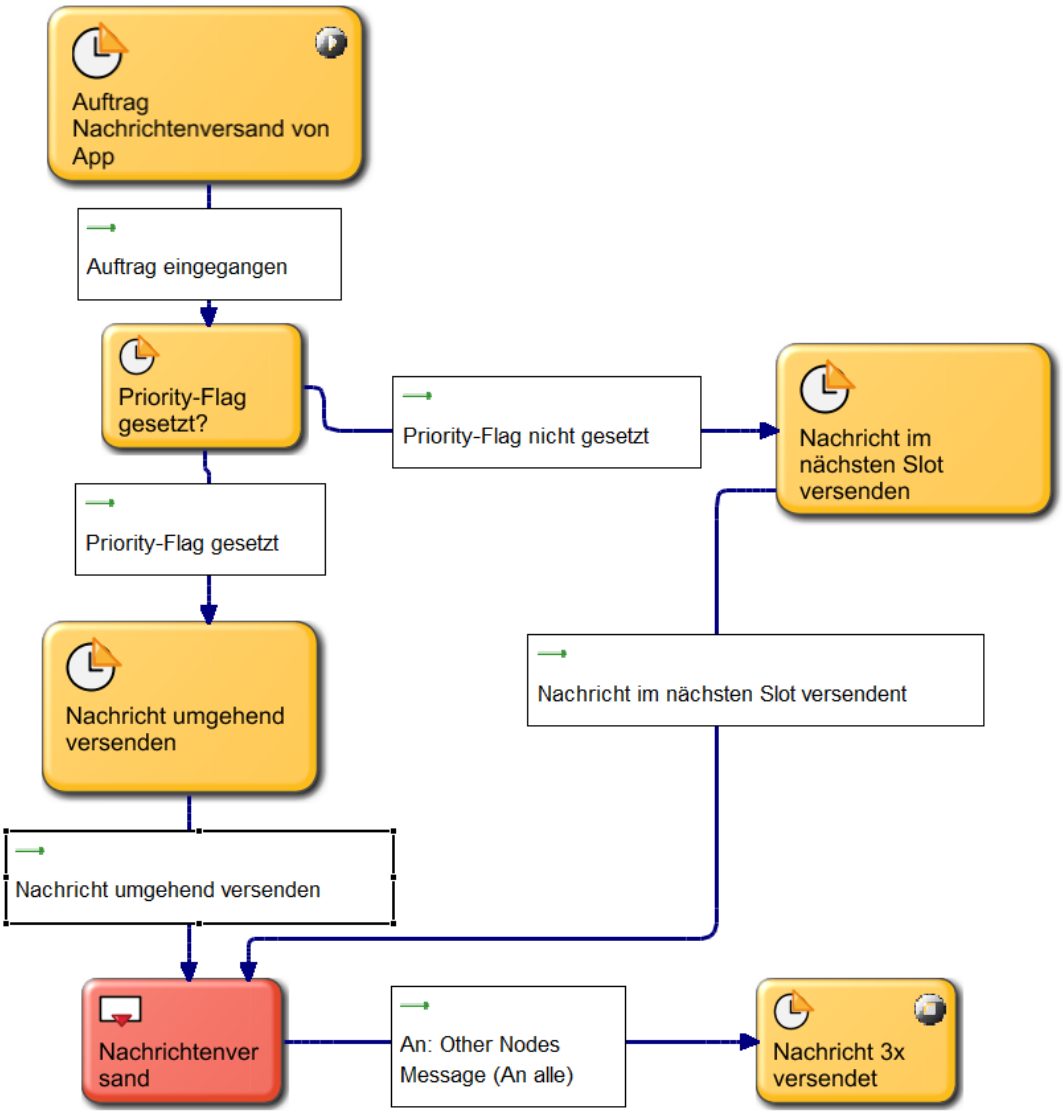


Abbildung 3.5: Interne Sicht des Own Nodes

Wenn das Netzwerkprotokoll den Auftrag zum Versand einer Nachricht bekommen hat, wird zu erst einmal überprüft, ob der Versand mit Priorität erfolgen soll oder die Nachricht im nächsten Versandslot mit versendet werden kann. In beiden Fällen wird die Nachricht aber als versendet markiert. Wie bereits erwähnt, bietet unsere Modellierungssuite nicht die Möglichkeit, den dreifachen Versand der Nachrichten darzustellen, das wird in

dem Prozess von dem Funktionszustand im letzten Punkt übernommen. Des Weiteren war es nicht möglich das implizite Acknowledgement für einzelne Knoten des Multisubjekts zu modellieren, bzw. zu testen, um bei Nichtempfang eventuell auch später nochmals die Nachricht auszustrahlen. In dem gezeigten Fall erhalten alle Knoten, die erreichbar sind korrekt die Nachricht, damit sind alle Einhop-Nachbarn gemeint.

Interne Ansicht der Other Nodes

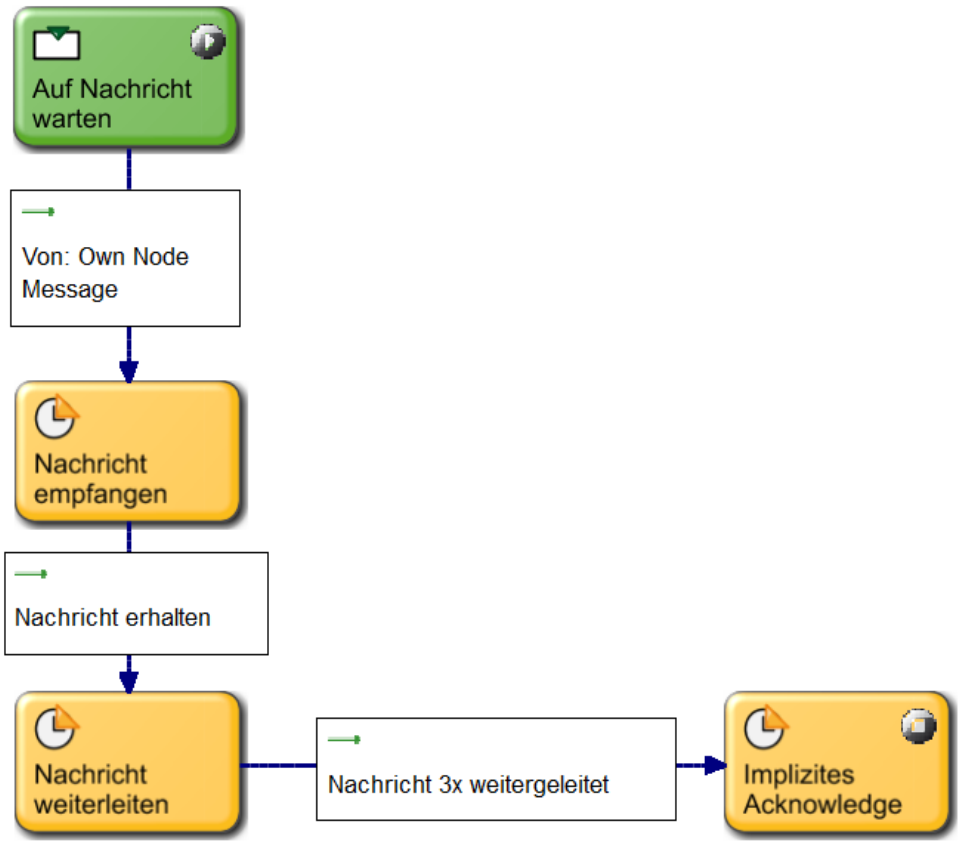


Abbildung 3.6: Interne Sicht der Other Nodes

Entsprechend einfach und robust ist auch das Empfängerverhalten modelliert. Wenn eine Nachricht empfangen wird, wird diese weitergeleitet, sofern diese im Gedächtnis des Knotens nicht bereits vorhanden ist, also nicht bereits drei Mal versendet wurde. Das beugt einem unbeschränkten Versenden der Nachrichten im Netzwerk vor. Auch dies

geschieht dreimal und dient dem ursprünglichen Versender der Nachricht als implizites Acknowledgement. Grundsätzlich ist der Prozess in diesem Schritt dem Versand des Own Nodes aus Abbildung 3.5 ähnlich, da auch hier die Checksumme in die entsprechende Tabelle aufgenommen wird.

3.5.2 Nachrichten-Relay

Nachfolgend soll die Skizze aus Abbildung 3.1 noch in einen S-BPM-Prozess überführt werden. Dabei wird für die interne Ansicht des Own Nodes auf die Abbildung 3.5 verwiesen, da der Versandprozess der ersten Nachricht identisch ist und deshalb wird nur die Kommunikationssicht und das interne Verhalten des Relays betrachtet.
Dieser Prozess zeigt exemplarisch die Kommunikation von verschiedenen Knoten auf, da wir hier einen grundsätzlichen Überblick über das Relaying geben wollen. Das Acknowledgement gilt immer jeweils für den vorherigen Knoten, als zum Beispiel von dem Bodenknoten zum AUV. Die Nachricht, die als Acknowledgement bezeichnet ist, soll das implizite Acknowledgement darstellen.

Kommunikationssicht

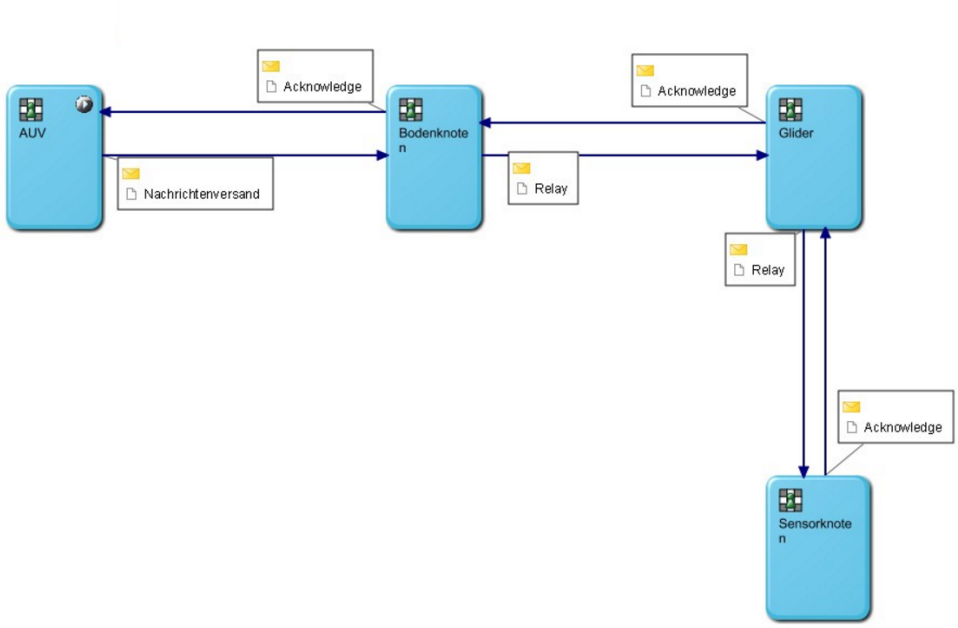


Abbildung 3.7: Kommunikationssicht beim Relay

Hier ist abgebildet, wie der Nachrichtenversand vom AUV bis zum Dreihop-Nachbarn ablaufen soll. Wie zuvor beschrieben, werden die Nachrichten jeweils weitergeleitet, das würde in der Realität jeweils in beide Richtung und nicht wie abgebildet in eine Richtung geschehen, wir haben das hier über das Acknowledgement versucht zu lösen.

Interne Ansicht des Relays

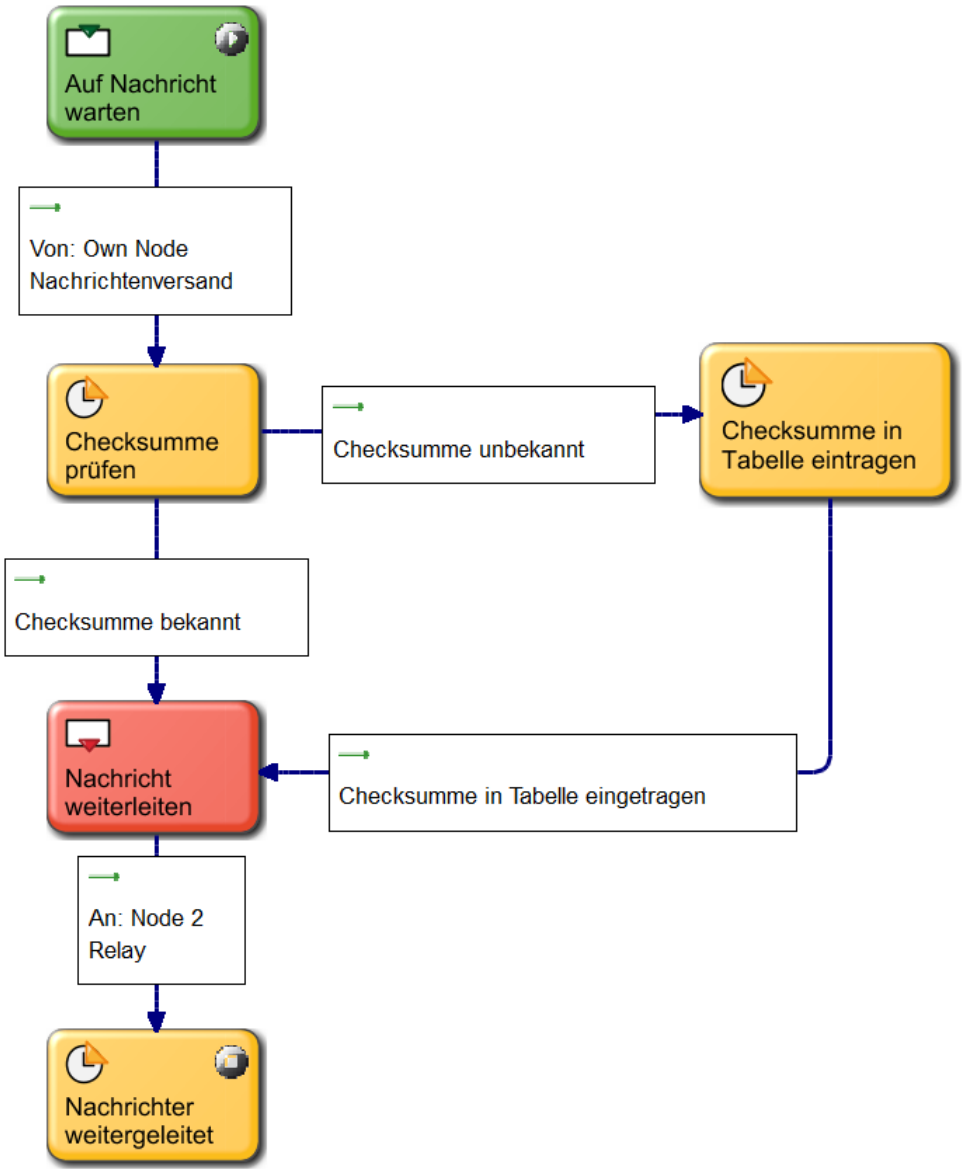


Abbildung 3.8: Interne Sicht des ersten Relays

Die Fähigkeit zu erkennen, ob eine Nachricht bereits erhalten wurde (Abbildung 3.6) benötigt das Gedächtnis. Da die Nachrichten sehr kurz sind, kann man die Nachrichten direkt vergleichen. Ansonsten kann man die Checksummen heranziehen.

3.6 Fehlertoleranz

Nachfolgend sollen die aufgeführten Komplikationen aus Abschnitt 3.3 aufgegriffen werden, viele Lösungsansätze sind bereits in Abschnitt 3.2 begründet.

Verteilte Systeme: Das Problem der verteilten Systeme lässt sich natürlich nicht umgehen, im Gegenteil, das Prinzip ist erwünscht bzw. erforderlich. Um trotzdem eine Kommunikation zu ermöglichen, versuchen wir durch das implizite Acknowledgement sicherzustellen, dass der Versender eine Rückmeldung über den Erhalt seiner Nachrichten bekommt und somit eine geschlossene Kommunikationen zwischen den Netzwerkknoten entsteht.

Adressierung: Unser Lösungsansatz ist hier in erster Linie die Checksumme der versendeten und erhaltenen Nachrichten in eine Look-Up-Tabelle einzutragen, um diese verfolgen und eine Zuordnung von Nachrichten zu den dazugehörigen Knoten ermöglichen zu können.

Netzwerküberlastung: Da wir davon ausgehen, dass die Netzwerkknoten generell bekannt sind und die Anzahl der Netzwerkteilnehmer unter Wasser überschaubar sein wird, halten wir es nicht für nötig, hier besonders zu agieren. Abgesehen davon, dass wir eigene Nachrichten nicht weiter relayen und somit Netzwerkknoten Nachrichten nur dreimal versenden sollten. Sollte ein häufigerer Versand gewünscht sein, müssen die Applikationen diesen Versand erneut anstoßen (siehe auch beim Postboten).

Nachrichten nicht erhalten oder nicht vollständig: Wie bereits erwähnt, werden unsere Nachrichten dreimal versendet, um eine Redundanz bieten zu können und so grundsätzlich gewährleisten zu können, dass die Nachrichten von anderen Knoten erhalten werden können. Sollte dies auf Grund von zu großen Distanzen nicht fehlerfrei möglich sein, besteht die Möglichkeit den Versand zu einem späteren Zeitpunkt erneut zu initiieren oder, falls möglich, die Position des Versenders oder des Empfängers zu verändern. Um erkennen zu können, ob eine Nachricht unvollständig ist, benutzen wir unsere Checksummen.

Fakenews: Dieser auch wahrscheinliche Fall lässt sich auf der Netzwerkebene abgesehen von der Checksumme, die bei Fakenews allerdings richtig sein sollte, nicht verhindern. Die Auswertung von Daten wird nicht auf Netzwerk- sondern auf der Applikationsebene übernommen. Ein semantischer Check ist auf der Applikationsebene notwendig.

3.7 Fazit & Ausblick

Wir haben in diesem Abschnitt unser Augenmerk auf die grundlegendsten Funktionalitäten gelegt, damit wir eine Schnittstelle zwischen den nachfolgenden Applikationen und dem Netzwerk bieten können. Diese sind anschließend in das S-BPM Modell überführt worden, um dies zu veranschaulichen. Leider bietet die Metasonic Suite nicht alle nötigen Funktionen, um den Prozessablauf mit mehreren gleichen Nachrichten und verschiedenen Empfängern abbilden zu können und dann entsprechend ein implizites Acknowledgement zu erhalten, bzw. zu versenden. Daher wurde versucht, die Prozesse möglichst abstrakt zu halten und zu beschreiben.
Eine dedizierte Adressierung in Form von Restricted Flooding hätten wir gerne noch implementiert, dies haben wir auf Grund der begrenzten Zeit leider nicht geschafft und uns deshalb auf das Fluten an sich fokussiert, das Gebiet des Restricted Floodings bietet Potenzial für weitere Forschung.

4 First Contact

Fatih Bülbül

4.1 Einleitung

Sensornetzwerke mit Kommunikationsknoten auf und unter dem Wasser, wie stationäre Bodenknoten, driftende Bojen, als auch mobile autonome Unterwasserfahrzeuge kommunizieren über Wasserschall, um Daten im Wasser zu übertragen. Der Austausch von Nachrichten erfordert häufig ein Zusammenspiel verschiedener Protokolle, die unterschiedliche Aufgaben übernehmen. Eine Aufgabe wäre der erste Kontakt. Damit ein erster Kontakt ordnungsgemäß zustande kommt, müssen die kommunizierenden Knoten dieselbe Protokollsprache nutzen, nämlich ein Standardprotokoll. Die Vereinbarung muss aus einem Satz von Regeln und Formaten bestehen (Syntaktische Interoperabilität), die das Kommunikationsverhalten der kommunizierenden Instanzen in den Knoten bestimmen (Semantische Interoperabilität). Außer der Nachrichtenübermittlung ist zu betrachten, ob die Nachricht richtig verstanden wurde. Diesbezüglich ist es wichtig, Fehlertoleranzen zu bilden, damit eine Nachricht, die möglicherweise syntaktisch richtig formuliert ist, aber semantisch falsch ist, am Ende trotz Komplikationen korrekt verstanden wird. Dieses Kapitel gibt einen Überblick über die syntaktische und semantische Interoperabilität, die mit dem First Contact verbunden sind und stellt mögliche Fehler, die auftreten können und dessen Behandlung dar.

4.1.1 Motivation

Die Unterwasserkommunikation ist sehr anspruchsvoll und vielfältig. Anwendungen werden mit der Zeit zunehmen, um unterschiedliche Aufgaben zu erfüllen. Mithilfe eines akustischen Unterwassernetzwerkes könnte man Wasserverschmutzungen entdecken, das Wetter präziser vorhersagen (z.B. Tsunamiwarnungen), den Klimawandel beobachten und Vorhersagen darüber treffen, wie das menschliche Handeln sich auf das Öko-System auswirkt. Außerdem ist die Mobilität des Netzwerkes sehr wichtig, wenn man überall auf der Welt die Möglichkeit der Kommunikation haben will. Das Problem ist, dass man Sen-

sorknoten nicht einfach in den gesamten Meeren der Welt verteilen kann, da die Kosten immens hoch wären und die Ressourcen womöglich dafür gar nicht ausreichen, alleine die Wartung wäre unmöglich. Mobile Knoten kann man von überall erreichen, diese sollen außerdem autonom und unbemannt ihre Aufgabe erfüllen, wie zum Beispiel das Absuchen eines Flugzeugwracks mithilfe von AUVs (Autonomous Underwater Vehicle). Die gesamte Kommunikation basiert auf dem Senden und Empfangen von Nachrichten (auf unterer Ebene). Wenn Subjekt A mit Subjekt B kommunizieren möchte, erstellt er zunächst eine Nachricht in seinem eigenen Adressraum. Dann führt er einen Systemaufruf durch, der das Betriebssystem veranlasst, die Nachricht über das Netzwerk an B zu senden. Obwohl diese grundlegende Idee einfach genug klingt, müssen sich A und B über die Bedeutung der übertragenden Bits verständigen, um Chaos zu vermeiden. Wenn A eine Nachricht auf französisch und im Zeichensatz ISO-8859-Text verschickt und B die Nachricht auf deutsch und in UTF-8-Umgebung kodiert erwartet, wird die Kommunikation unmöglich optimal verlaufen. Derzeit existieren mehrere Dutzend Modulations- und Netzwerkprotokolle. Damit besteht der Bedarf einer automatisierten Absprache unter Wasser.
Es werden viele verschiede Vereinbarungen benötigt. Woher weiß der Empfänger, was die Bits der Nachrichten bedeuten? Wie kann er erkennen, ob eine Nachricht beschädigt wurde oder verloren gegangen ist, und was muss er tun, wenn er das herausfindet? Kurz gesagt, es sind Vereinbarungen auf einer Vielzahl von Schichten erforderlich. Mit JANUS gibt es ein First Contact Protokoll ähnlich zum Channel 16 im Marine-Funk, maschinell muss man sich das wie eine Zeichensprache auf unterstem Niveau vorstellen. Ein Kommunikationsknoten hat zu anderen Knoten unterschiedliche Ressourcen. Das Netz muss diese Ressourcen erlernen. Das sind Art der Modulation, Kodierung, Mapping, Frequenzbereiche usw..

4.1.2 JANUS als Standardprotokoll

JANUS ist ein Erst-Kontakt-Protokoll mit einer 64-bit-Paketstruktur für den Austausch von Daten zwischen Subjekten, die in einem Unterwassernetzwerk miteinander verbunden sind. JANUS ist unter zwei Grundsätzen entwickelt. Der erste Grundsatz besagt Minimal-Anforderungen für maximale Interoperabilität, also stellt es eine nahtlose Zusammenarbeit zwischen den Systemen, einen effizienten und verwertbaren Informationsaustausch bereit, ohne dass dazu eine gesonderte Absprache zwischen den Systemen notwendig ist. Gemäß dem zweiten Grundsatz ist JANUS ein offener Standard (public domain). Damit ist das Protokoll frei von Ansprüchen und für alle zugänglich, benutzbar und editierbar. Dementsprechend besitzt das Protokoll 256 Anwenderklassen und bietet jedem Anwender die Möglichkeit, 64 Anwendungen selber zu definieren. Die meisten An-

wenderklassen bestehen aus allen Ländern der Welt und die restlichen sind bestimmte Klassen, z.B. für den Notfall. Die Anwendungen sind noch nicht vollständig ausdefiniert - es fehlt der First Contact und ein Teil soll in dieser Arbeit vorgeschlagen werden. Die Datenübertragungsrate beträgt 16-40 bit/s und die Datenmenge ist 26-64 bit groß, also dauert eine Übertragung von 64 bit mindestens 1,6 s. Das JANUS Protokoll ist eigentlich nicht für Netzwerke oder Datentransfer geeignet, jedoch für Command und Control, Warnung und Kleinstdaten nutzbar. In diesem Fall kann man sich an den Ablauf der Kommunikation im Channel 16 halten. Eine kurze Einführung dazu soll einen Einstieg bzw. ein Verständnis für den First Contact unter Wasser aufbauen.

4.1.3 „Channel 16“ unter Wasser

Im maritimen Bereich nutzt man den Channel 16 um z.B. ein Mayday zu versenden und zu empfangen, dieser Channel wird 24 h von den Küstenwachen weltweit überwacht und soll hauptsächlich nur im Notfall genutzt werden. Ausschließlich ist es nur der Küstenwache erlaubt kurze Sicherheitsinformationen abzuschicken. Dieses Verfahren soll verhindern, dass es zu einer Überflutung von Nachrichten kommt und dadurch wichtige Nachrichten überhört werden. Außerdem findet nach einer erfolgreichen Absprache ein Kanalwechsel statt. Diese Vorgehensweise verdeutlicht, dass der Channel 16 hauptsächlich für den Verbindungsaufbau genutzt wird, damit eine weitergehende Kommunikation über einen anderen Kanal folgen kann.
Die aufgezählten charakteristischen Eigenschaften des Verfahrens lassen sich ebenso auf die Systeme unter Wasser übertragen. Sender und Empfänger richten explizit eine Verbindung ein, bevor sie Daten austauschen und handeln möglicherweise dazu verwendete Protokolle aus.

4.1.4 Ausblick

In diesem Teil des Projekts werden die Prozessabläufe für den First Contact hergeleitet. Diese Prozessabläufe werden anfangs an einem Beispiel mit Bezug auf die reale Welt anschaulich dargestellt. Darauffolgend wird auf eine mögliche Komplikation im Prozessablauf eingegangen, welches mit Hilfe einer Skizze zunächst gelöst wird. Danach werden die Nachrichten in JANUS definiert und beschrieben. JANUS stellt ein Application Data Block von 34 bit bereit, welches dem Nutzer erlaubt, selbst definierte Nachrichten festzulegen. Im Anschluss daran folgen die Modelle der Prozesse in *Subject-oriented Business Process Modelling* (S-BPM). Diese Art der Modellierung stellt Funktionen von Subjekten und deren Kommunikation dar. Eine detaillierte Erklärung über S-BPM ist in Kapitel

2.3 aufzufinden. Zuletzt wird auf mögliche Fehlerarten in den Prozessen eingegangen und denkbare Lösungen mit Bezug auf das Modell vorgestellt. Fehlertoleranzen sind wichtig, um Prozesse aufrechtzuerhalten, wenn unvorhergesehene Eingaben oder Fehler in der Hard- oder Software auftreten. Außerdem werden auch byzantinische Fehler ermittelt. Diese liegen dann vor, wenn ein Knoten eine falsche Nachricht an einen oder mehreren Knoten sendet. Unter falscher Nachricht wird hierbei eine korrekte Syntax aber ein Fehler in der Semantik der Nachricht verstanden.

4.2 Erste Prozessformulierung

Vor der detaillierteren Ansicht in S-BPM werden Beurteilungen und die Ausführung über den ersten Kontakt in einem Schema zusammengefasst dargestellt. Dies soll einen groben Überblick über den gesamten Prozess darstellen, welches in Abbildung 4.1 zu sehen ist. Um einen vernüftigen Kontakt herstellen zu können, muss der Empfänger zunächst eine Nachricht empfangen, welche für ihn verständlich sein muss. Demzufolge wird vorausgesetzt, dass es eine Standardsprache gibt. Die Kommunikation findet in diesem Kapitel über natürliche Sprachen statt. Des Weiteren braucht der Empfänger relevante Informationen, um den Absender zu identifizieren:

- Name
- Nachbarknoten
- Haupt-/ Nebensprache

Der Name dient dazu, um den Knoten zu identifizieren und persönlich anzusprechen. Des Weiteren werden Informationen über die bekannten Knoten des Absenders gegeben. Zuletzt sendet der Absender noch Informationen über seine Haupt- und Nebensprache. Dabei gehen wir vom besten Fall aus, also hat der Prozess folgende Voraussetzungen: Die Namen der Knoten sind unterschiedlich, die Hauptsprache ist übereinstimmend und die Nachrichten kommen korrekt an.
Der Knoten A sendet in Abbildung 4.1 (a) als Fremder die erste Nachricht ab, um sich bei den unbekannten Knoten registrieren zu können. Die erste Nachricht wird auf englisch definiert und enthält wichtige Informationen, über die Hauptsprache und Nebensprache, außerdem die Namen der bekannten Knoten und seinen eigenen Namen A. In der Nachricht wird angegeben, welche Muttersprache er beherrscht, hierbei wäre es Deutsch und die Nebensprache Spanisch. In Abbildung 4.1 (a) ist das Rechteck der unbekannte

Knoten C, der nun die empfangene Nachricht registriert und die Informationen über die Sprachen und den bekannten Knoten mit seinen eigenen auf eine Übereinstimmung überprüft. Da wir als Voraussetzung auf eine übereinstimmende Hauptsprache gesetzt haben, antwortet der Knoten C in Abbildung 4.1 (b) auf Deutsch mit seinem eigenen Namen und die Namen der Knoten, die er kennt. Nun hat Knoten A einen persönlichen Kontakt zu den Knoten C und B, letztere ist in diese Situation C aufgrund der Frequenzbänder nicht bekannt gewesen. Außer den Namen weiß nun A, ob die Haupt- oder Nebensprachen übereinstimmt oder nicht. Dahingegen weiß er nicht, ob die Sprachen von Knoten D und E mit seinen übereinstimmen. Also kennt Knoten A die zwei Knoten nur unter ihrem Namen und nicht mehr. Man definiert sie hierbei als *two-hop neighbor*.

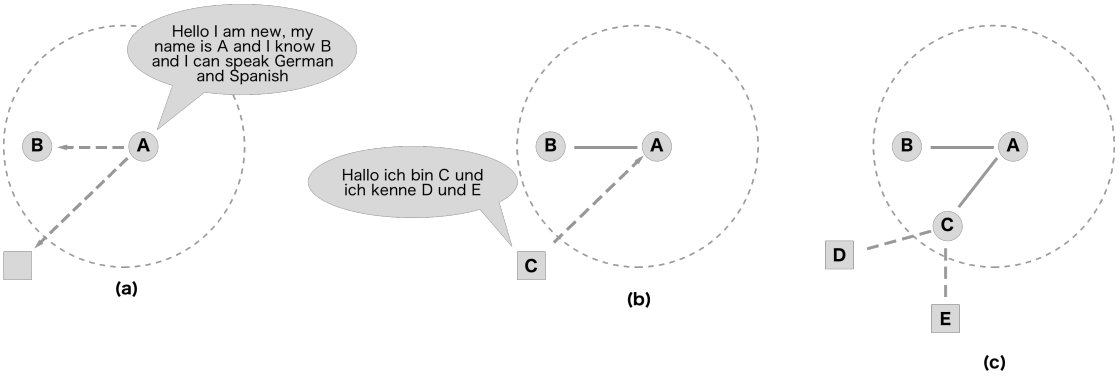


Abbildung 4.1: Prozess des ersten Kontakts. (a) A sendet als Fremder die erste Kontaktnachricht. (b) C versteht die von A genannte Sprache und antwortet auf die Nachricht von A. (c) A und C kennen sich gegenseitig.

4.3 Komplikationen bei First Contact

Im besten Fall hatten wir eine funktionierende und übereinstimmende Kommunikation. Dies muss aber nicht immer der Fall sein, da die Kommunikation auf einem höheren Niveau behindert werden kann, wenn beide kommunizierenden Knoten unterschiedliche Haupt- und Nebensprachen beherrschen. Nun bräuchte man einen Übersetzer, der über Kenntnisse in beiden Sprachen verfügt, die Fähigkeit, beide Sprachen miteinander zu verbinden und perfektes Wissen der Terminologie. Er sollte niemals versuchen, seine eigenen Ideen im Zieltext einzufügen. Sondern das Ziel haben, den Inhalt und die Intention des Quelltextes so exakt wie möglich im Zieltext wiederzugeben.

Der Übersetzer wird dann eingesetzt, wenn ein Empfänger-Knoten aus der Nachricht vom fremden Absender keine Übereinstimmung der Sprachen erkennt. Nun würde er sich

beim Dolmetscher melden. Dieses Konzept wird in Abbildung 4.2 veranschaulicht dargestellt. Wie in dem letzten Kapitel dargestellten Szenario, ist der Knoten A der fremde Knoten, der neue Knoten kennenlernen möchte. Dieser sendet seine erste Kontaktnachricht ab und erhält nun die Nachricht von Knoten C, dass er keine kompatiblen Sprachen beherrscht, sondern die Sprachen Französisch und Italienisch (Abbildung 4.2 (b)). Nun hat Knoten B beide Nachrichten erhalten und erkennt, dass er beide Hauptsprachen der anderen Knoten beherrscht. Dieser meldet sich dann bei den anderen Knoten als Dolmetscher (Abbildung 4.2 (c).

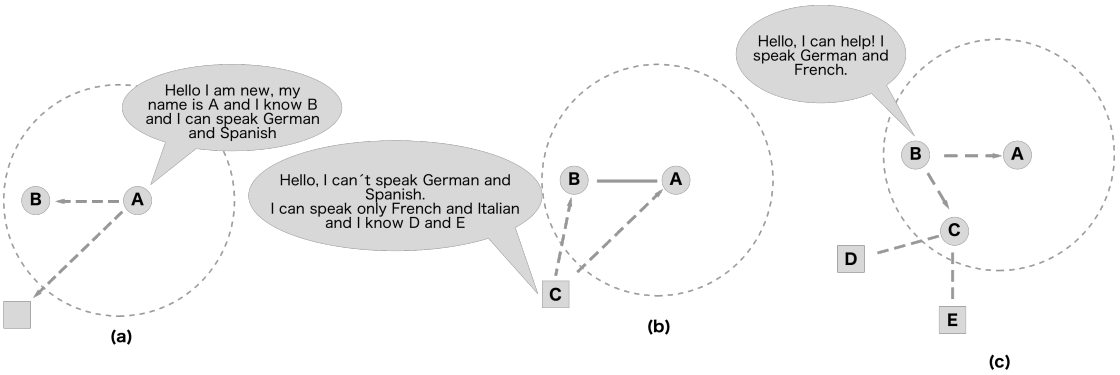


Abbildung 4.2: Einsatz eines Dolmetschers. (a) A sendet als Fremder die erste Kontaktnachricht. (b) C kann die von A genannten Sprachen nicht sprechen und antwortet auf die Nachricht von A. (c) B stellt fest, dass er die Hauptsprachen der Knoten beherrscht und meldet sich als Dolmetscher.

4.4 Nachrichten in First Contact

Bevor die internen Prozesse der S-BPM-Modelle dargestellt werden, wird zunächst die Kommunikationssicht der S-BPM-Modellierung in Abbildung 4.3 beschrieben, um einen Überblick zu verschaffen. Danach gibt es eine Einführung über das *Open Systems Interconnection Model* (OSI Modell), um anschließend zu verstehen, wie eine Modulationskonfiguration in einer Nachricht gebildet werden muss.

4.4.1 Kommunikationssicht in S-BPM

Es gibt drei Subjekte ownNode (ich), otherNode (Fremder) und als neue Erweiterung der Dolmetscher translatorNode. Es gibt drei Telegramm-Päckchen, diese sind in JANUS definiert. Wenn beide Knoten dieselben Konfigurationen besitzen, hierbei wird das GUWAL-Paket als Beispiel genommen, dann werden nur noch in der übereinstimmenden Konfiguration Nachrichten ausgetauscht.

Für den ersten Kontakt wurde eine neue Anwenderklasse in JANUS definiert, bei der nun die Class user i.d. auf den Wert 4 gesetzt wird (siehe Abbildung 3.2, Bits 9-16). Die Nachrichtentypen sind fest in den Anwendungen definiert, es sind 64 Anwendungen pro Anwender möglich. Jede Anwendung kann man als eine Telegramm-Nachricht sehen.
In der ersten Anwendung ist eine Nachricht für die Bekanntgebung des eigenen Namen, die Namen der benachbarten Knoten und die eigene Modulationskonfiguration enthalten. In der Kommunikationssicht sendet otherNode die erste Nachricht aus dem JANUS Protokoll „Hello, I am the new one“ ab. Nun muss ownNode die Nachricht auf Richtigkeit überprüfen. In diesem Fall erstellt er eine Checksumme aus dem Payload und vergleicht diese mit der gesendeten Checksumme von otherNode. Damit gibt es drei Zustände, die Nachricht ist gültig, fehlerhaft oder es gibt keine Nachricht. Falls die Nachricht gültig ist und die Modulationskonfigurationen übereinstimmen, antwortet er mit einer GUWAL-Nachricht.
Falls die Modulationsverfahren nicht verwendet werden können, dann wird eine JANUS-Nachricht gesendet und auf einen Dolmetscher gehofft. Diesbezüglich gibt es eine weitere Anwendung: „Hello, I am not compatible“. Wenn ein translatorNode die Nachrichten von otherNode und die von ownNode erhält und über beide Konfigurationen verfügt, dann meldet er sich beim otherNode mit einer GUWAL-Nachricht und beim ownNode mit einer JANUS-Nachricht „Hello, I can help“. Nun würde der translatorNode die Aufgabe übernehmen zwischen beiden Knoten selbständig zu übersetzen.

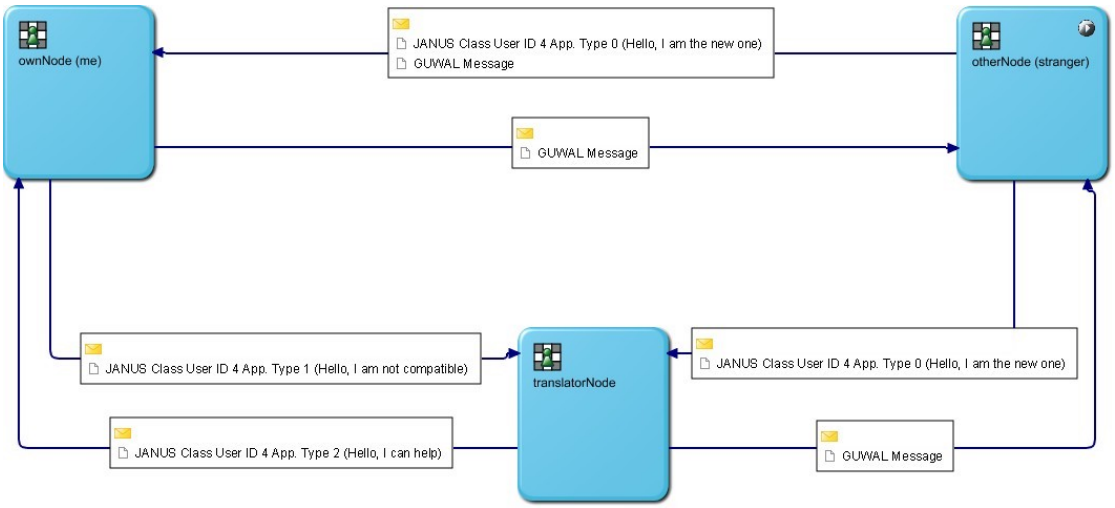


Abbildung 4.3: Kommunikationssicht des First Contacts

Die Nachrichten „I am not compatible“ und „I can help“ treffen ebenfalls auch bei OtherNode ein, wie auch die zu übersetzenden Nachrichten mittels GUWAL, die über

den translatorNode ausgetauscht werden. Dieser Ablauf ist damit komplexer als hier dargestellt.

4.4.2 OSI-Modell

Um eine Kommunikation zwischen offenen Systemen zu ermöglichen, wurde das OSI-Modell entworfen. Dies erfolgt nur unter Verwendung von Standardregeln, die das Format, den Inhalt und die Bedeutung der gesendeten und empfangenen Nachrichten bestimmen. Diese Regeln werden in sogenannten Protokollen formalisiert. Um einer Gruppe von Systemen die Kommunikation über ein Netzwerk zu ermöglichen, müssen sich alle mit den verwendeten Protokolle verständigen.

Im OSI-Modell wird die Kommunikation auf sieben Schichten oder Ebenen aufgeteilt, wie in Abbildung 4.4 gezeigt. Jede Schicht behandelt einen bestimmten Aspekt der Kommunikation. Auf diese Weise kann das Problem in handliche Teile zerlegt werden, die alle unabhängig voneinander gelöst werden können. Wenn das Subjekt A auf dem Rechner 1 mit dem Subjekt B auf dem Rechner 2 kommunizieren möchte, erstellt er eine Nachricht und übergibt sie der Anwendungsschicht auf seinem Rechner. Die Software der Anwendungsschicht fügt dann vor der Nachricht eine Kopfzeile (Header) ein und übertragt die sich so ergebende Gesamtnachricht über die Schnittstelle an die nächste Schicht. Diese fügt ebenso ihren eigene Kopfzeile hinzu und übergibt das Ergebnis nach unten an die nächste Schicht. Dieser Vorgang geht so lange weiter, bis die letzte Schicht, die Bitübertragungsschicht erreicht worden ist. In dieser wird die Nachricht tatsächlich übermittelt, in dem sie sie dem physischen Transportmedium übergibt.
Wenn die Nachricht bei Rechner 2 ankommt, wird sie nach oben übermittelt, wobei jede Schicht ihren eigene Kopfzeile entfernt und auswertet. Schließlich erreicht sie den Empfänger, Subjekt B, der auf sie antworten kann, indem er den umgekehrten Weg verwendet. Die Informationen in der Kopfzeile von Schicht n werden für das Protokoll auf Schicht n verwendet.

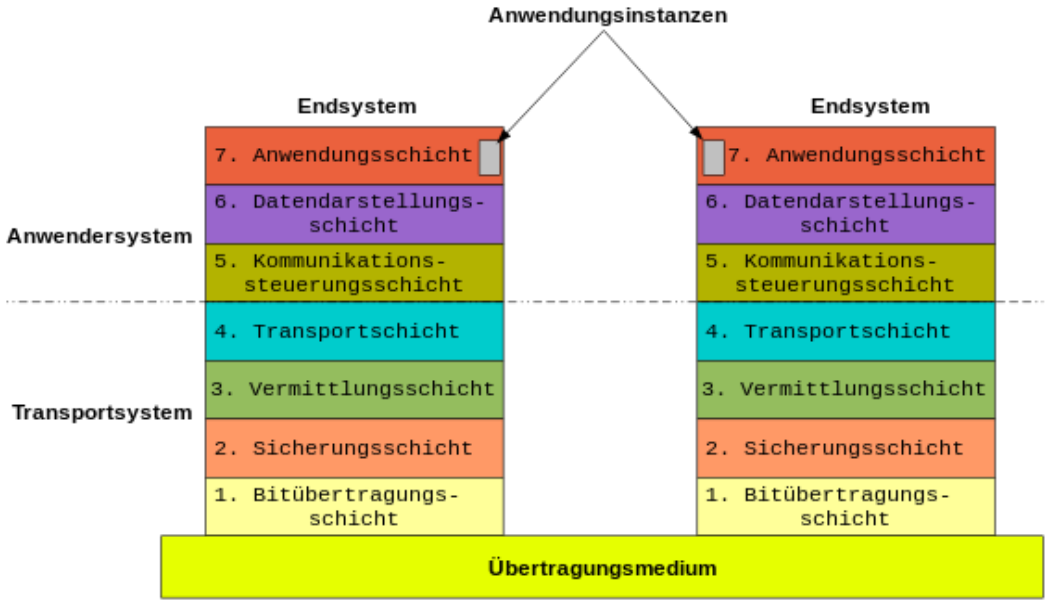


Abbildung 4.4: ISO-OSI-7-Schichten-Modell (aus Deadlyhappen [2014])

Als Beispiel für die Bedeutung von Schichtprotokollen wird eine Kommunikation vorgestellt und zwar zwischen zwei älteren Personen, die voneinander sehr weit weg wohnen und noch über die altmodische Art ihre Nachrichten senden, nämlich per Post. Dabei nutzen beide ihre Enkel für die Übermittlung der Nachricht. Nach einer bestimmten Zeit wird es für die Enkel zu mühselig, die Postkarte bei der Post abzugeben. Dabei entscheiden sie sich beide für die zukünftige Kommunikation per E-Mail die Nachrichten zu senden. Sie konnten diese Entscheidung ohne Rücksprache mit ihren Großvätern treffen, da ihr Protokoll die physische Übertragung der Nachricht betrifft und nicht deren Inhalt. Entscheidend ist, dass hier zwei Schichten betrachtet werden: Der Großvater und der Enkel. Jede Schicht hat ihr eigenes Protokoll, das unabhängig vom anderen ausgetauscht werden kann, ohne dass die anderen davon beeinflusst werden.

4.4.3 JANUS-Nachrichten

Die JANUS-Anwendung erlaubt eine Liste mit 16 verschiedenen Angaben über die Phy (Physical Layer) Konfiguration, 8 Angaben über die Net (Network Layer) Konfiguration. Zuletzt kann es auch 8 Angaben über das Kanalmodel machen.

PhyConfiguration (modulation and coding)			
0	MFSK	1 of 4 coding	LF
1	MFSK	hadamard coding	LF
2	MFSK	1 of 4 coding	HF
3	MFSK	hadamard coding	HF
4	OFDM	rate 1/2	VLF
5	OFDM	rate 2/3	VLF
6	FRSS		VLF
7	FRSS		HF
8	DSSS		VLF
		...	
14	Burst		VLF
15	nothing		

Tabelle 4.1: Modulation und Coding

NetConfiguration (link and networklayer)	
0	GUWMANET
1	GUWMANET+
2	dflood
3	SeaWeb
...	
7	nothing

Tabelle 4.2: Link und Networklayer

ChannelModel	
0	WSSUS
1	no WSS/ US
2	WSS/ no US
3	no WSS/ no USS
4	fast fading
...	
7	no measurement

Tabelle 4.3: Channel Model

Das JANUS-Protokoll bietet ohne folgenden CARGO-Container höchstens 34 bit als Payload an. Die Anwendung (Application type) gibt die zugehörige Telegramm-Nachricht an. Wenn diese festgelegt ist und ein Knoten, beispielsweise ownNode die JANUS-Nachricht 0 sendet, dann weiß der Empfänger, um was es in der Nachricht geht. Wenn konkret

die Nachricht 0 „Hello, I am the new one“ (Tabelle 4.4) beispielsweise von otherNode empfangen wird, sind damit Name, Nachbarschaft, Konfigurationen und CRC für den Bezug (als Key) von ownNode bekannt. Der Name ist 4 bit groß, dementsprechend sind Angaben bis zu 16 Namen, die dem otherNode bekannt gemacht werden, möglich. Diesbezüglich wurde in dem Payload ein 16 bit großes Boolean-Array deklariert. Falls der Inhalt ein true enthält, dann ist der Name belegt. Ansonsten ist der Name bei false unbekannt. Wenn alle false sind, dann sind keine Nachbarn bekannt. Die letzten 14 bits in dem Payload werden von der Kommunikationskonfiguration belegt.
Die Nachricht 1 „Hello, I am not compatibel“ (Tabelle 4.5) enthält außer den Namen und der Konfiguration, den CRC-Link der Nachricht aus Tabelle 4.4 und zwei Nachbarnamen. Ist kein weiterer Knoten gegenüber der Nachricht message 0 bekannt, werden otherNeighborName1 und 2 mit ownName belegt. Diese Informationen soll ein Übersetzer-Knoten erhalten. Der Dolmetscher enthält in seiner Nachricht (Tabelle 4.6) den eigenen Namen, den CRC-Link aus der Nachricht der Tabelle 4.5 und die Konfigurationen von otherNode (Tabelle 4.4) und ownNode (Tabelle 4.5). Damit soll bekannt gegeben werden, dass der Translator über beide Konfigurationen Nachrichten austauschen kann.

Szenario: "Hello, I am the new one"	
Class User ID = 4	
Application type = 0	
buildInPayload JANUS := [(34bit)	
- ownName (last nibble of name)	4 bit
- knownNeighbors (my neighborhood) array[16] of boolean (true, falls Name belegt/ bekannt, false unbekannt, alle false, keine Nachbarn bekannt)	16 bit
- mainPhyConfiguration (modulation and coding)	4 bit -
secondPhyConfiguration (modulation and coding)	- 4 bit
NetConfiguration (link and networklayer)	- 3 bit
ChannelModel	3 bit
]	
Checksum	- 8 bit
CargoHold :=[]	

Tabelle 4.4: JANUS-Nachrichtenprotokoll: Nachricht 0 ("Hello I am the new one")

Szenario: "Hello, I am not compatible!"	
Class User ID = 4	
Application type = 1	
buildInPayload JANUS := [(34bit)	
- ownName (last nibble of name)	4 bit
- your CRC-Link (CRC of message 0)	8 bit
- otherNeighborName1 (current)	4 bit
- otherNeighborName2 (second current)	4 bit
- mainPhyConfiguration (modulation and coding)	4 bit
- secondPhyConfiguration (modulation and coding)	4 bit
- NetConfiguration (link and networklayer)	3 bit
- ChannelModel	3 bit
]	
Checksum	8 bit
CargoHold :=[]	

Tabelle 4.5: JANUS-Nachrichtenprotokoll: Nachricht 1 ("Hello, I am not compatible!)

Szenario: "Hello, I can help"(Dolmetscher, Translator)	
Class User ID = 4	
Application type = 2	
buildInPayload JANUS := [(34bit)	
- ownName (last nibble of name)	4 bit
- CRC-Link (CRC of message 1)	8 bit
- LanguageMessage 0, siehe PhyConfiguration)	4 bit
- LanguageMessage 1, siehe PhyConfiguration)	4 bit
- NetConfiguration (link and networklayer)	3 bit
- ChannelModel	3 bit
]	
Checksum	8 bit
CargoHold :=[]	

Tabelle 4.6: JANUS-Nachrichtenprotokoll: Nachricht 2 ("Hello, I can help")

4.5 Modellierung in S-BPM

Es werden zwei First Contact Ereignisse betrachtet, welche schon in den vorherigen Kapiteln aufgegriffen wurden. Beim ersten Ereignis findet ein direkter Kontakt und beim zweiten ein indirekter Kontakt statt.

otherNode → ownNode: otherNode findet einen Knoten, dessen Modulationskonfiguration übereinstimmt. Für den ersten Kontakt wird die JANUS-Nachricht „Hello, I am the new one“ (Tabelle 4.4) verwendet.

otherNode → ownNode/translatorNode → ownNode/otherNode: Die Modulationskonfiguration von ownNode stimmt nicht mit der von otherNode überein, jedoch besitzt ein anderer Knoten (translatorNode) diese Konfiguration. ownNode wendet sich an einen translatorNode, der die Nachrichten für die beiden Knoten in situ übersetzt.

Der Prozess beginnt mit otherNode in Abbildung 4.5. otherNode ist das fremde Subjekt und möchte sich in der Umgebung bei den Knoten vorstellen. In der *Applikationsebene* (Application Layer (APP)) wird die Nachricht „Hello, I am the new one“ erzeugt. Diese enthält Informationen über den Namen, die Nachbarnamen und über die Konfiguration. Eine detaillierte Ansicht der Nachricht ist in Tabelle 4.4 zu sehen. Danach wird es auf der *Bitübertragungsschicht* (Pyhsical Layer (PHY)) gesendet und darauf gehofft, dass ein ownNode diese erhält. Nun wird auf eine Antwort gewartet. Die Wartezeit beträgt zwischen 5-15 Minuten. Der Grund für eine Wartezeit ist, dass die Übertragungsgeschwindigkeit unter Wasser nicht so hoch ist und der Empfänger weit entfernt sein kann. Außerdem kann es vorkommen, dass die Nachricht niemand erhält, wenn es an Empfängern in der Umgebung fehlt oder die Nachricht selber beschädigt ist. Eine beschädigte Nachricht wird über die *Checksumme* erkannt. Diese wird in jeder Nachricht mit gesendet und mit der Checksumme, die der Empfänger generiert, verglichen. Falls innerhalb der Wartezeit keine Antwort kommt, wird die Nachricht erneut gesendet. Dies geschieht höchstens drei Mal mit der Einhaltung der Wartezeit. Falls immer noch keine Antwort kommt, dann wird die Position geändert und die Nachricht nochmals gesendet. Wenn die Positionsänderung effektlos war, dann kann es vorkommen, dass das System eine unwahre Nachricht sendet, dementsprechend startet das System neu und erstellt die Nachricht erneut. Nun wird von Beginn an versucht ownNode eine Nachricht zu senden. Im Extremfall, wenn das System überhaupt keine Antworten bekommt und ein Neustart auch vergebens war, dann fährt der mobile Knoten zur Basis zurück und lässt sich Warten.
Falls eine Antwort empfangen wird, ist dies eine Bestätigung, dass die Nachricht angekommen ist und außerdem kann man anhand des Nachrichtentyps erkennen, ob die Konfigurationen übereinstimmen. Weiterhin muss betrachtet werden, ob die Nachricht des Absenders syntaktisch gültig ist, welches mit der Überprüfung der Checksumme geschieht. Falls es nicht der Fall ist, wird die Nachricht auf der *Vermittlungsebene* (Network

Layer (NET)) erneut gesendet. Wenn die Nachricht syntaktisch gültig ist und die Konfigurationen übereinstimmen, dann werden nur noch mit dieser die Nachrichten gesendet.

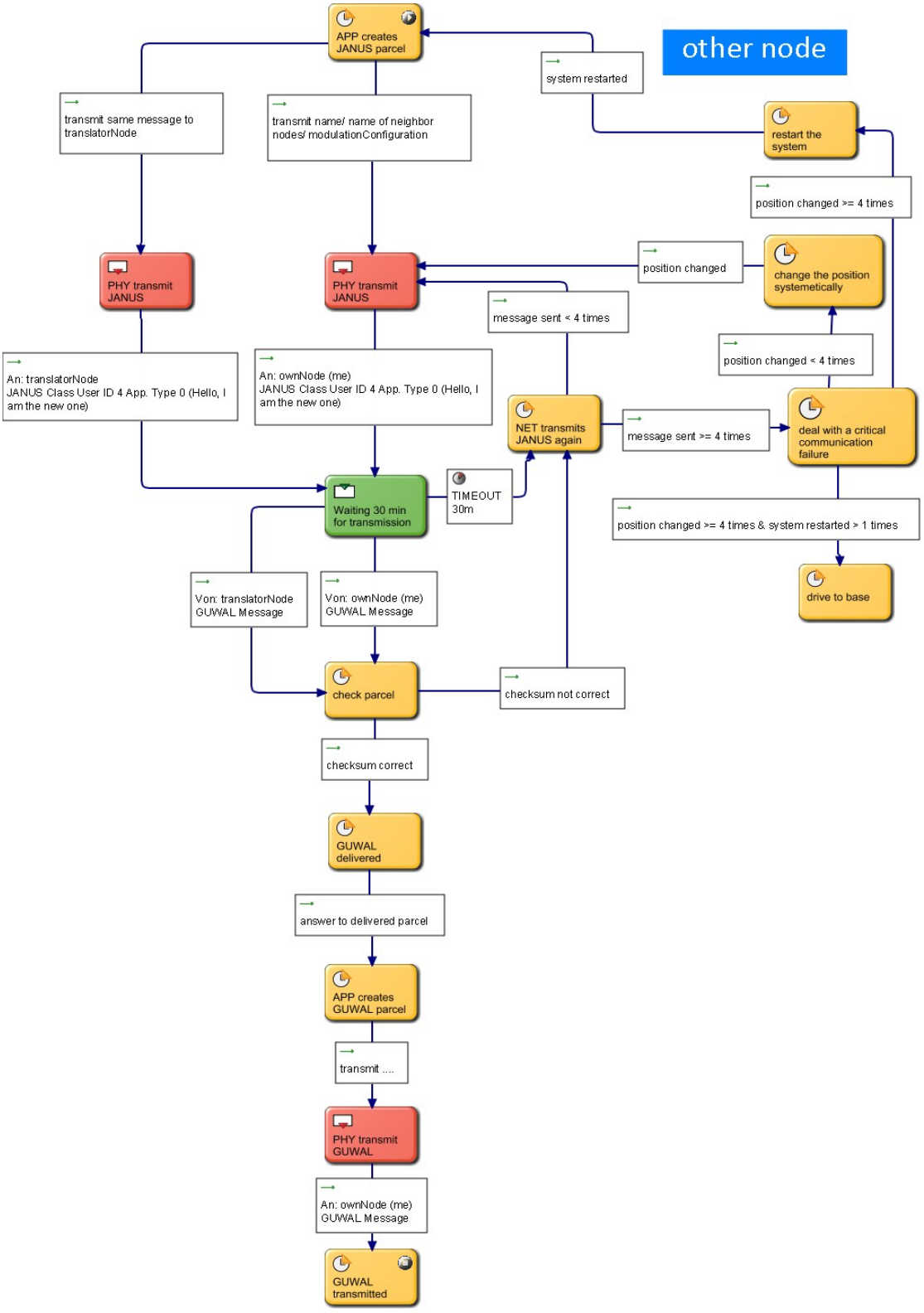


Abbildung 4.5: Internes Verhalten eines unbekannten Knotens

ownNode wartet in Abbildung 4.6 auf eine Übertragung, konkreter auf die Nachricht von otherNode aus Abbildung 4.5. Wenn die erste Nachricht von otherNode angekommen ist, prüft ownNode über die Checksumme die Gültigkeit des Pakets. Wenn diese nicht korrekt ist, dann wechselt der Knoten wieder auf den Empfangszustand. Bei Gültigkeit prüft er nun die Nachricht, ob diese übereinstimmende Konfigurationen enthält.
Falls eine Übereinstimmung (Abbildung 4.7) da ist, wird auf der Applikationsebene ein GUWAL-Paket erstellt, welches mehr Payload bereitstellt als das JANUS-Paket. In dieser können außer den Namen, Nachbarnamen und der zyklischen Redundanzprüfung (cyclic redundancy check (CRC)) der ersten Nachricht weitere Informationen über den Standort oder den Energiezustand enthalten sein. Auf der Bitübertragungsschicht wird dann das Paket gesendet. Wie in den vorherigen Modell, wird auf eine Antwort im Minutenbereich gewartet und das Paket bei einer Rückmeldung auf Gültigkeit überprüft. Somit wäre dann der Fall der ersten Kommunikation für ownNode beendet.

Falls es keine Übereinstimmung gibt, wird in der Applikationsebene ein JANUS-Paket erstellt. Dieser enthält die Nachricht „Hello, I am not compatible" (Tabelle 4.5). Mit dieser Nachricht versucht man einen Übersetzer dazu zu holen, der für beide Seiten übersetzt. Wenn es einen Knoten gibt, der für beide Seite übersetzt, dann antwortet er mit der JANUS-Nachricht „Hello, I can help" (Tabelle 4.6).

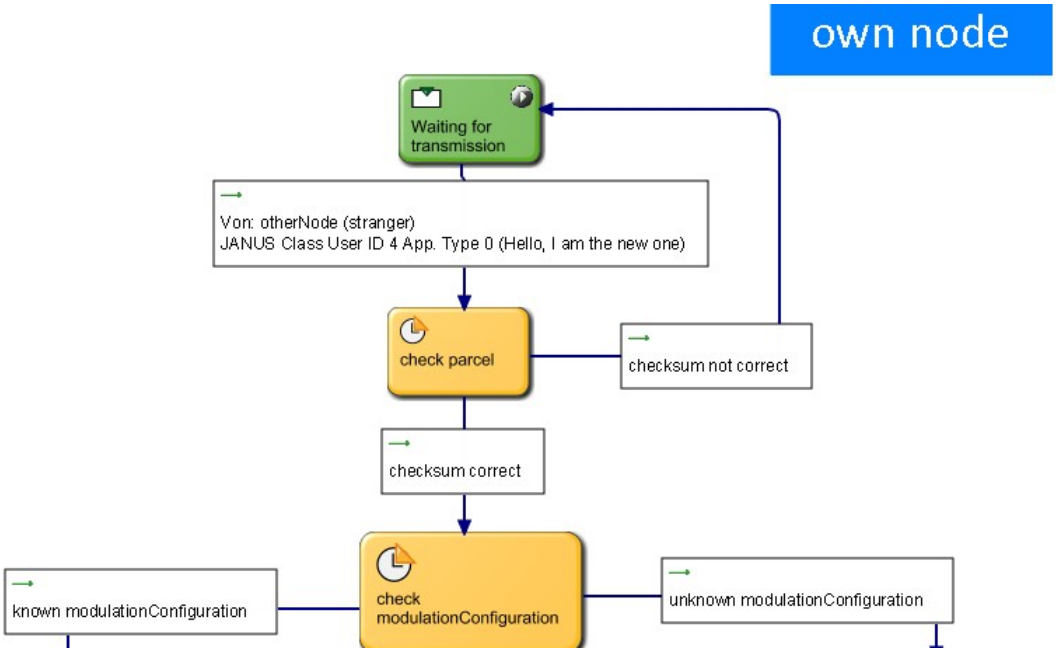


Abbildung 4.6: Internes Verhalten von ownNode.

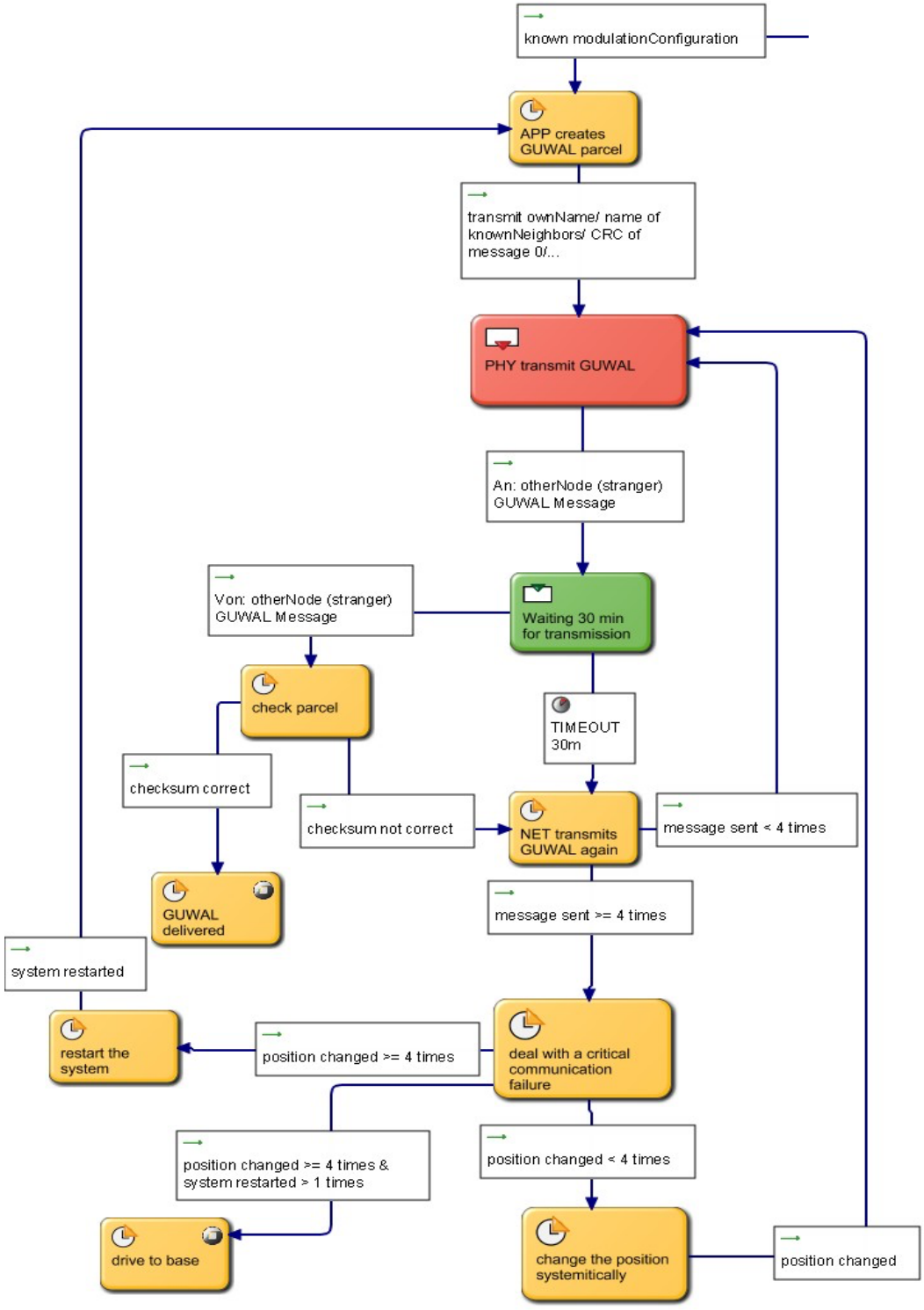


Abbildung 4.7: Internes Verhalten von ownNode bei bekannter Konfiguration.

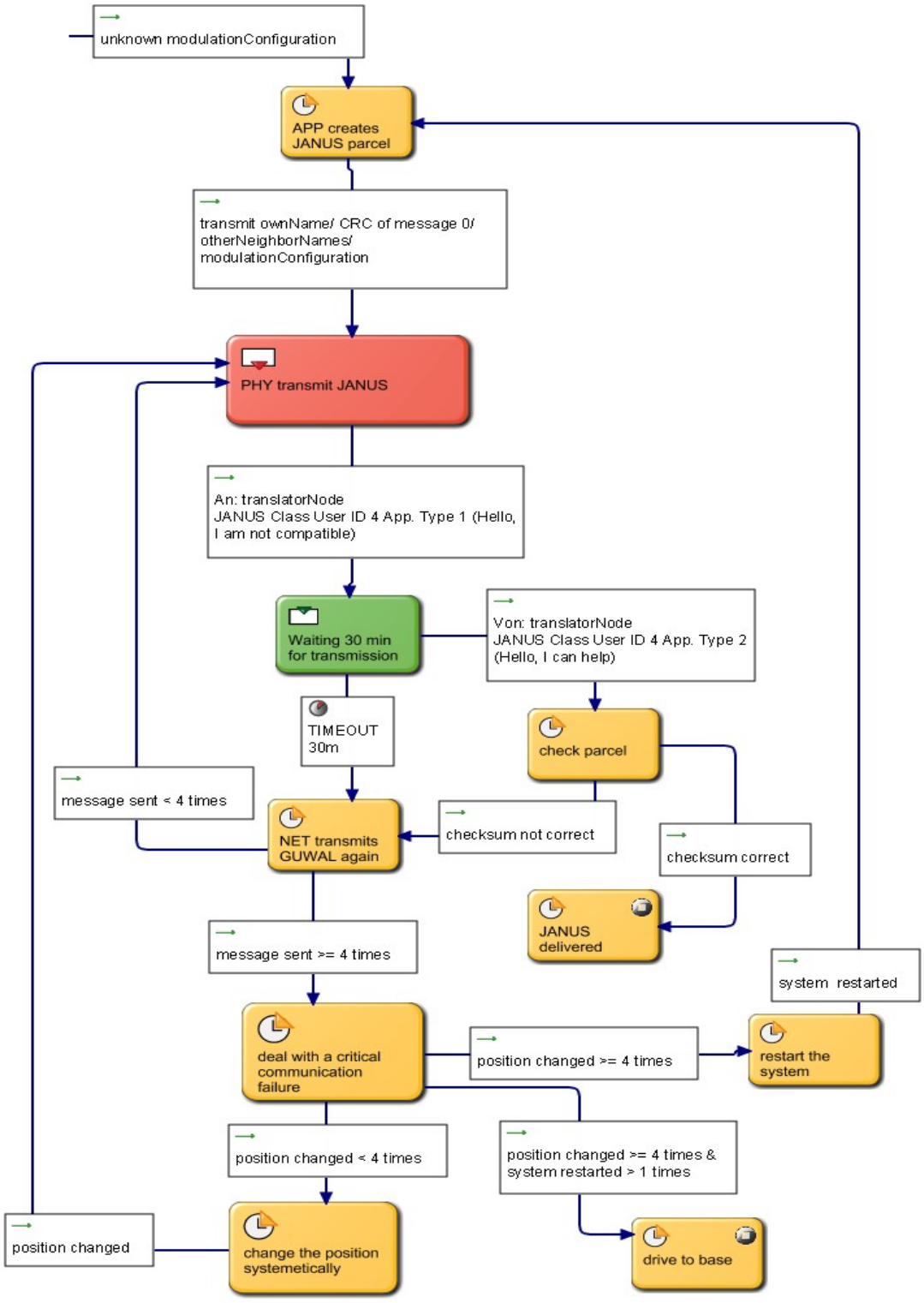
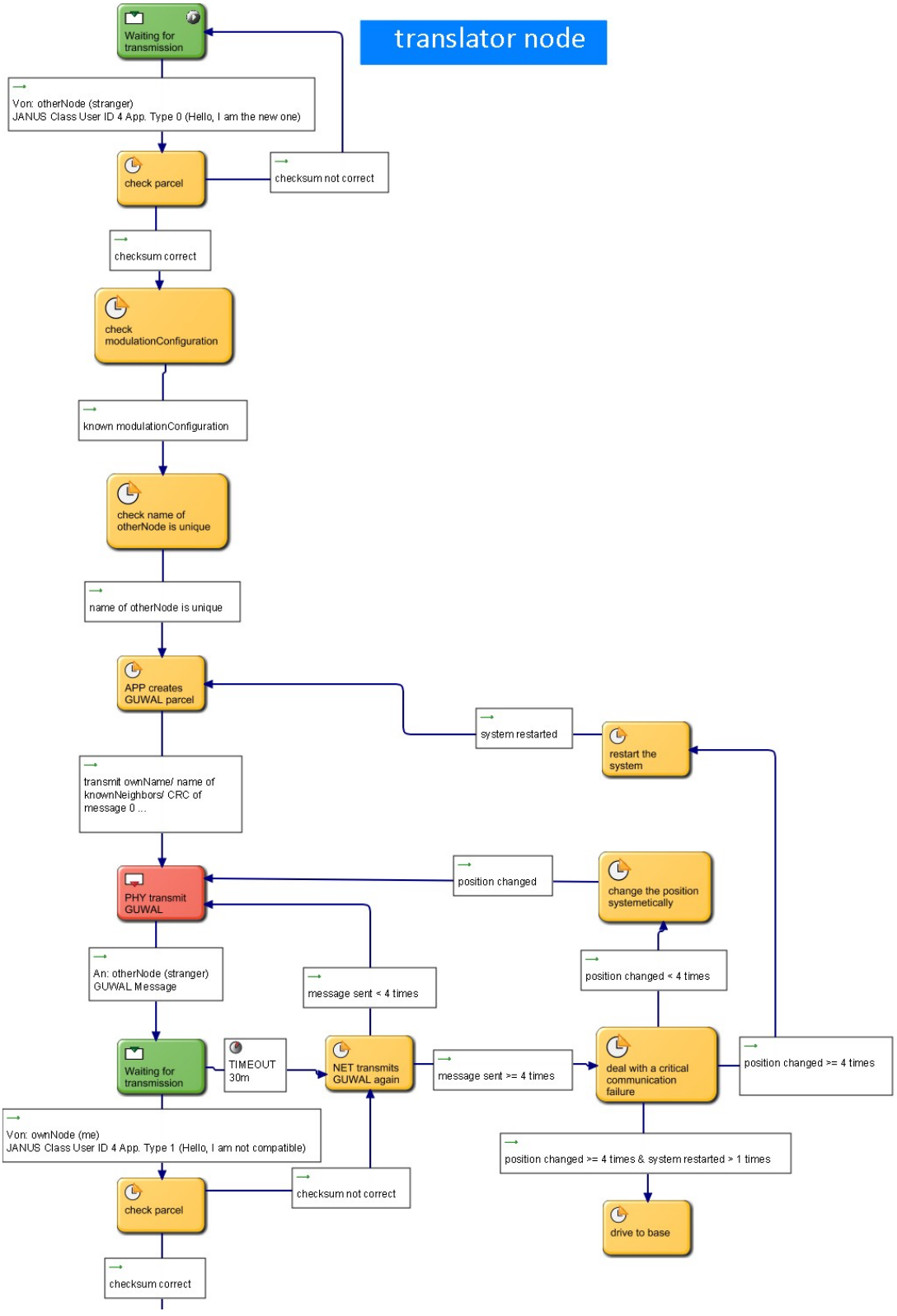


Abbildung 4.8: Internes Verhalten von ownNode bei unbekannter Konfiguration.

Der translatorNode empfängt wie der ownNode (Abbildung 4.6) eine JANUS-Nachricht „Hello, I am the new one" (Tabelle 4.4). Wenn die Nachricht syntaktisch gültig ist und die Modulationskonfiguration übereinstimmt, dann antwortet er mit einem GUWAL-Paket. Das Verfahren ist wie bei ownNode. Nun geht es wieder auf den Empfangszustand über und erhält die Nachricht von ownNode „Hello, I am not compatible". Nun wird in der Applikationsebene das JANUS-Paket für die Nachricht „Hello, I can help" erstellt. In dieser sind Informationen über die Modulationskonfiguration der ersten Nachricht von otherNode und der zweite Nachricht von ownNode. Damit wäre der Prozess für den translatorNode abgeschlossen und kann jetzt als Übersetzer fungieren.


translator node
Waiting for transmission
Von: otherNode (stranger)
JANUS Class User ID 4 App. Type 0 (Hello, I am the new one)
check parcel
checksum not correct
checksum correct
check modulationConfiguration
known modulationConfiguration
check name of otherNode is unique
name of otherNode is unique
APP creates GUWAL parcel
system restarted
restart the system
transmit ownName/ name of knownNeighbors/ CRC of message 0 ...
PHY transmit GUWAL
position changed
change the position systemetically
An: otherNode (stranger)
GUWAL Message
message sent < 4 times
position changed < 4 times
Waiting for transmission
TIMEOUT 30m
NET transmits GUWAL again
message sent >= 4 times
deal with a critical communication failure
position changed >= 4 times
Von: ownNode (me)
JANUS Class User ID 4 App. Type 1 (Hello, I am not compatible)
position changed >= 4 times & system restarted > 1 times
check parcel
checksum not correct
drive to base
checksum correct

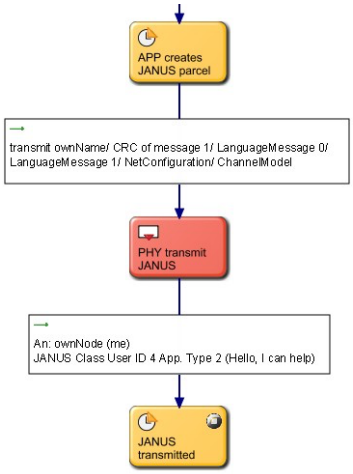


Abbildung 4.9: Einsatz eines Dolmetschers.

4.6 Fehlertoleranzen

4.6.1 Fehlerbehandlung in modellierten Prozessen

Hier sollen mittels Beispiele betrachtet werden, wie die modellierten Prozesse auf Fehler reagieren.

Nachricht ist beschädigt

Potentieller Fehler Der Absender sendet seine Nachricht und wartet auf eine Anwort. Die Nachricht die eintrifft ist beschädigt und wird von dem Empfänger mit der Prüfsumme auf diese Fehlerhaftigkeit untersucht.

Modellierte Lösung Die Bitübertragungsschicht sendet lediglich bits. Solange kein Fehler passiert, ist alles in Ordnung. In realen Kommunikationsnetzwerken treten jedoch Fehler auf, sodass ein Verfahren benötigt wird, um sie zu entdecken und zu korrigieren. Dieses Verfahren ist die Hauptaufgabe der Sicherungsschicht. Es gruppiert die bits in Einheiten, manchmal Rahmen (Frames) genannt und achtet darauf, dass jeder Rahmen korrekt empfangen wird. Die Sicherungsschicht fügt zur Markierung ein besonderes Bitmuster an den Anfang und das Ende eines jeden Rahmens an und berechnet eine Prüfsumme durch eine Form der Addition aller im Rahmen enthaltenen Bytes. Sie fügt dem Rahmen die Prüfsumme hinzu. Wenn der Rahmen ankommt, berechnet der Empfänger die Prüfsumme aus den Daten neu und vergleicht das Ergebnis mit der dem Rahmen folgenden Prüfsumme. Stimmen sie nicht überein, bittet der Empfänger den Sender um

die erneute Übertragung. Rahmen erhalten (im Kopf) laufende Nummern, sodass jeder erkennen kann, wer welcher ist.

Nachrichtenübermittlung ist fehlgeschlagen

Potentieller Fehler Die Nachrichtenübermittlung kann fehlschlagen.

Modellierte Lösung Es wird ein Timeout von mehreren Minuten eingesetzt. Wenn innerhalb dieser Zeit keine Bestätigungsnachricht ankommt, wird die Nachricht erneut auf der Netzwerkebene übermittelt. Dieser Vorgang wiederholt sich bei Überschreitung der Minutenzeiträume höchstens drei Mal und toleriert den Verlust von Nachrichten.

OtherNode erhält keine Antwort

Potentieller Fehler OtherNode sendet eine Nachricht, um eine Verbindung mit einem anderen Knoten aufzubauen und eine Entscheidung über eine Konfiguration zu treffen. Jedoch bekommt otherNode keine Antwort bei dreimaligen Senden. Außerdem bringt das mehrfache Senden aus verschiedenen Positionen auch nichts. Es könnten verschiedene Fehlerarten bei otherNode, aber auch bei den anderen Knoten vorliegen. Eine mögliche Fehlerart könnte eine Empangauslassung (Receive ommision) beim otherNode sein. Eine andere Fehlerart könnte bei den anderen Knoten liegen wie ein Dienstausfall (Omission Failure). Ebenfalls kann ein Fall vorliegen, dass es kein ownNode gibt. Dieser Fall wird in dieser Arbeit nicht betrachtet und kann in weiteren Arbeiten ausformuliert werden.

Modellierte Lösung In Abbildung 4.5 wird in so einem Fall der Prozess neugestartet und es wird von Beginn an versucht, eine Verbindung aufzubauen. Wenn dies auch nicht funktionieren sollte, fährt der Knoten zur Basis zurück und ein Mensch repariert das System.

4.6.2 Nicht behandelte Fehler

In diesem Teil der Arbeit wird auf mögliche Fehler eingegangen, die in dem Modell nicht behandelt wurden.

Namenskonflikt

Potentieller Fehler In der Nachricht wird ein Name angegeben, den man bereits kennt. Mögliche Behandlungsstrategien finden sich in der Bachelorarbeit von Mike Buschhorn.

Nachricht enthält die falsche Modulationskonfiguration des Absenders

Potentieller Fehler Die Einträge im Protokoll ergeben keinen Sinn. Es wird ein Modulationsverfahren angegeben, welches der Sender nicht spricht. Hierbei handelt es sich um byzantinische Fehler. Die Nachricht ist syntaktisch korrekt, ebenso die Angaben über die Modulationskonfiguration, jedoch ist diese semantisch falsch. In dieser Arbeit erstellten Prozess würde der Empfänger Knoten die falsche Modulationskonfiguration hinnehmen. Hierbei würde er sich entweder für eine Übereinstimmung entscheiden oder sich bei keiner Übersetzung beim Übersetzer-Knoten melden. D.h in der Applikationsebene wird entweder mit der Konfiguration aus der Nachricht des Absenders eine Nachricht erstellt oder es wird erneut mit einer JANUS-Nachricht geantwortet. Es muss also in dem Prozess etwas eingebettet werden, damit das System in so einem Fall nicht falsch reagiert.

Mögliche Lösung In den aufgezählten Fällen kann der Empfänger Knoten keine richtige Verbindung mit dem Absender aufbauen. Nun muss es trotz der korrekten Syntax der Nachricht ein Verfahren geben, welches diesen Vorfall erkennt und behandelt. Denn das Problem könnte Konsequenzen mit sich bringen, sobald die falsche Modulationskonfiguration mit der des Empfängers übereinstimmt. Somit würde es keine Gegenantwort geben, wenn der Empfänger mit dieser Konfiguration antwortet. Dies bedeutet er würde annehmen, dass der Fehler bei ihm liegt, also dass er vielleicht eine Empfangsauslassung hat oder seine Nachricht eventuell nicht gelesen werden kann. In diesem Fall würde der Knoten neustarten und sehen, dass das erneute Senden nach dem Neustart auch nichts bringt. Damit würde er sich sich wieder zur Basis begeben und sich reparieren lassen wollen. So ein Verhalten hätte gravierende Auswirkungen, wenn nur ein Knoten alle Systeme auf dieser Art und Weise stören könnte. Demzufolge wird ein Verfahren gebraucht, welches genau so etwas verhindern soll. Das Verfahren der Fehlerbehandlung muss auf jeden Fall beim Empfänger stattfinden. Nun nehmen wir an, dass der Absender eine falsche Modulationskonfiguration gesendet hat und der Empfänger über eine Übereinstimmung besitzt. Der Empfänger versucht nun dem Absender z.B. mit einer GUWAL-Nachricht zu erreichen. Er wiederholt seine Nachricht höchstens drei Mal und versucht aus anderen Positionen seine Nachricht zu übermitteln, jedoch wird es wie vorhin erklärt fehlschlagen. Nun könnte eine JANUS-Nachricht erstellt werden, der den Absender benachrichtigt, dass der Empänger ihn nicht mit der Konfiguration erreichen kann. In der Nachricht kann die Modulationskonfiguration und die CRC des Absenders enthalten sein und die eigene Modulationskonfiguration. Wenn der Absender nun Bescheid gegeben bekommt, dass er die Modulationskonfiguration falsch sendet, dann kann er sich dafür entscheiden entweder neu zu starten und erneut versuchen den Knoten zu erreichen. Außerdem kann

der Absender nun entscheiden, ob er seine Modulationskonfiguration richtig sendet oder nicht, denn es kann vorkommen, dass nicht der Absender die Schuld an der fehlgeschlagenen Nachrichtenübermittlung hat. Wie man sehen kann, steckt in so einem Prozess die Gefahr, dass ein ganzes Netzwerk damit ausfallen kann. Dementsprechend sollten vorerst wichtige Überlegungen darüber getroffen werden, wie man solche Fehler behandelt.

4.7 Fazit & Ausblick

Die Unterwasserkommunikation ist nicht ausgereift, gewinnt jedoch an Bedeutung. Zurzeit gibt es keine effiziente Lösung für den Konfigurationsaustausch, da dies aktuell manuell geschieht und viel Zeit kostet. Eine Lösung wäre das System zu autonomisieren, sodass problemlos eine Verbindung ohne einen Eingriff von Außen ermöglicht werden kann. Essentiell bei so einer Lösung ist die Toleranz gegenüber Verbindungsfehlern.

Der Hauptkern dieser Ausarbeitung liegt in der Vorstellung von Konzepten, die als Basis für ein autonomes Unterwassernetzwerk dienen könnten. Zur Einführung wurden das Protokoll JANUS vorgestellt und ein Beispiel aus der realen Welt benutzt, um die Prozessabläufe besser zu verstehen. Als nächstes wurden der Inhalt der Nachricht bestimmt und der Ablauf des Prozesses mit Hilfe einer Skizze veranschaulicht. Daraufhin wurde auf eine wesentliche Komplikation eingegangen, bei der der Empfänger und der Absender keine übereinstimmende Konfiguration enthält. Das Problem lies sich mithilfe eines Übersetzers lösen. Die Verwendung des JANUS-Protokolls ermöglicht den Inhalt der Nachricht als Bitfolge zu kodieren. Der Prozess wurde danach in S-BPM modelliert und die definierten Nachrichten den Subjekten zugeordnet und ausgeführt. Außerdem behandelt das Modell ein Teil der möglichen Fehler, die detaillierter in dem letzten Abschnitt beschrieben wurde. Schließlich kommen auch mögliche Lösungen für nicht behandelte Fehler vor. Die eingeführten Modelle in dieser Arbeit dienen als Basis für mögliche Lösungsansätze für die Kommunikation unter Wasser. Anhand dieser kann nach weiteren Fehlern gesucht werden.

5 Distance Measurement

Kim Wachlin

5.1 Einleitung

In diesem Themenbereich geht es um die Messung der Streckendistanz zwischen zwei oder mehreren Subjekten unter Wasser. Dabei sollen entsprechende Prozesse modelliert werden, die sich grundsätzlich in zwei verschiedene Bereiche unterteilen lassen:

- Distanzmessung zwischen genau zwei Subjekten
- Distanzmessung zwischen mehr als zwei Subjekten

Diese Prozesse werden in Hinblick auf ihre Fehlertoleranz analysiert und auf Methoden der Fehlerelimination überprüft.

5.1.1 Motivation und Problematik

Die Distanz ist beispielsweise erforderlich für den Informationsaustausch, denn ein Subjekt muss unter Wasser wissen, wie weit es sich einem Kommunikationsteilnehmer noch nähern muss, um eine bessere Übertragungsrate für den Austausch der Daten zu erlangen. Das Problem dabei ist, dass der naive Ansatz des Satelliten-GPS nicht funktioniert, da die elektromagnetischen Wellen nach einer nur sehr kurzen Strecke im Wasser „verschluckt“ werden. Stattdessen muss die Distanz mittels Schallwellen ermittelt werden. Jedoch variiert die Schallgeschwindigkeit unter Wasser. Sie ist abhängig von der Temperatur, der Dichte und dem Salzgehalt des Wassers. Daher ist es bei der Distanzmessung zwischen mehreren Subjekten notwendig, dass die Uhren dieser Subjekte synchronisiert sind. Allerdings sind die Uhren mit einem nicht zu vernachlässigen Zeitdrift versehen und die Subjekte können sich aufgrund der Schallabsorption nicht mit den Satelliten synchronisieren. Dafür wird ein Synchronisierungsmechanismus genutzt, mithilfe dessen sich die Subjekte selbst synchronisieren können.
Für die Distanzmessung zwischen zwei Subjekten hingegen ist keine Zeitsynchronisation vonnöten, da hierbei lediglich relative Zeiten genutzt werden.

Durch die Absorption der GPS-Wellen funktioniert natürlich auch die herkömmliche GPS-Ortung nicht. Daraus ergibt sich ein weiterer wichtiger Anwendungsfall der Distanzmessung: die Positionsbestimmung mittels Multilateration. Diese kann genutzt werden, wenn die Distanzen zu mindestens drei Subjekten und deren Position bekannt ist.

5.1.2 Ausblick

In diesem Teil der Ausarbeitung werden Prozessabläufe für die Distanzmessung zwischen zwei Subjekten und für die Distanzmessung zwischen mehr als zwei Subjekten am Beispiel einer Positionsbestimmung erklärt und modelliert, wobei der Fokus hierbei auf fehlerhaften Nachrichten liegt. Das heißt, es werden Mechanismen entworfen, wie Fehler gehandhabt, toleriert oder behandelt werden können, ohne dass sich der Prozess aufhängt und somit das System abstürzt oder nicht ordnungsgemäß läuft. Fehler können hierbei unterschiedlicher Natur sein, zum Beispiel kommt eine erwartete Nachricht nicht fristgerecht an oder enthält - trotz korrektem Aufbau und Checksum - nicht verwertbare oder unschlüssige Informationen. Die Modellierung erfolgt mittels bereits in Kapitel 2.3 vorgestellter Modellierungsnotation *S-BPM*. Für die Nachrichten werden die Protokolle *JANUS* (Kapitel 3.2) und *GUWAL* (Kapitel 3.4) verwendet.

5.2 Prozessformulierung

In diesem Abschnitt werden die zu modellierenden Prozesse informal vorgestellt. Dies erfolgt durch eine Einführung in die beiden in der Einleitung erwähnten Hauptprozesse in natürlicher Sprache mithilfe von Skizzen. Im späteren Verlauf der Ausarbeitung sollen diese Prozessformulierungen dann genutzt werden, um sie zur Modellierung in *S-BPM* auszuweiten.

5.2.1 Distanzmessung zwischen zwei Subjekten

Die Distanzmessung zwischen zwei Subjekten ist der einfachere Ansatz, da hierbei keine Synchronisation benötigt wird, sondern stattdessen relative Zeiten verwendet werden. Die hier verwendete Methode nennt sich *mark-snap* (NATO [2002]) und funktioniert wie folgt:

- Subjekt A sendet *„mark“*
- Subjekt B sendet direkt beim Empfang *„snap“*
- Subjekt A sendet direkt beim Empfang von *„snap“* erneut *„mark“*

Abbildung 5.1 veranschaulicht dieses Szenario.

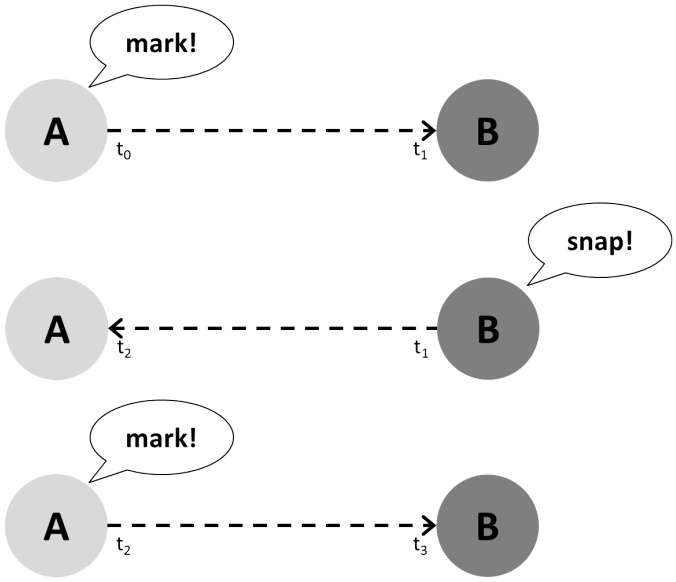


Abbildung 5.1: Distanzmessung mithilfe der *mark-snap*-Methode

Dadurch können nun beide Subjekte bei gegebener Schallgeschwindigkeit die Distanz zueinander bestimmen. Man nimmt dafür die Zeitspanne zwischen Senden und Empfangen, subtrahiert die Bearbeitungszeit des Kommunikationspartners und halbiert das Ergebnis. Dadurch erhält man die Übertragungsdauer und kann nun mithilfe der Schallgeschwindigkeit die Distanz bestimmen. Da diese unter Wasser variiert, nehmen wir hierbei eine durchschnittliche Schallgeschwindigkeit von $c = 1500\,\frac{m}{s}$ an. Formal sieht das wie folgt aus:
Sei

$\mathrm{s} \mathrel{\widehat{=}}$ gesuchte Distanz,

$\mathrm{t_t} \mathrel{\widehat{=}}$ Zeitspanne zwischen Senden und Empfangen ($t_2 - t_0$ respektive $t_3 - t_1$),

$\mathrm{t_p} \mathrel{\widehat{=}}$ Bearbeitungszeit des Kommunikationspartners.

Dann ergibt sich für die Übertragungsdauer $\frac{t_\mathrm{t} - t_\mathrm{p}}{2}$ und somit gilt nach dem Weg-Zeit-Gesetz für die Distanz

$$s = \frac{t_\mathrm{t} - t_\mathrm{p}}{2} * c$$

Diese Vorgehensweise ist nur ein einfacher Ansatz, der keine genauen Messwerte liefert. So wird hierbei von einer ungefähren Schallgeschwindigkeit ausgegangen. Diese variiert, wie bereits erwähnt, jedoch unter Wasser. Weiterhin kann diese Methode lediglich Distanzen zwischen zwei beteiligten Subjekten bestimmen. Für viele Anwendungsfälle, wie zum Beispiel Positionsbestimmungen im dreidimensionalen Raum, benötigt man jedoch mindestens drei Distanzen. Ein denkbarer Anwendungsfall dieser Methode wäre die Entfernungsschätzung zu einem Subjekt, mit dem Daten ausgetauscht werden sollen. Denn bestimmte Übertragungsmethoden können erst ab einer bestimmten Nähe verwendet werden, dies ist insbesondere hilfreich beim Übertragen großer Datenmengen. Ein weiterer Anwendungsfall wäre die Einschätzung, ob der gewünschte Zielort mit vorhandenem Akkustand noch erreicht werden kann. Für diese beiden Szenarien reicht ein ungefährer Richtwert aus, es müssen keine detaillierten Distanzen bestimmt werden. Daher reicht auch die gerundete Schallgeschwindkeit aus.

5.2.2 Distanzmessung zwischen mehr als zwei Subjekten

Da sich der Schall nicht mit konstanter Geschwindigkeit und nicht linear ausbreitet, zum Beispiel am Boden reflektiert wird, wird von allen beteiligten Subjekten die genaue Uhrzeit benötigt. Allerdings sind die digitalen Zeitsysteme mit einem Zeitdrift von mehreren

Sekunden in nur wenigen Tagen versehen. Da der Schall etwa 1500 $\frac{m}{s}$ zurücklegt, würde dies zu größeren Messfehlern führen. Aus diesem Grund benötigt man ein zeitsynchronisiertes Netzwerk mit einer gemeinsamen Zeitbasis. In diesem Abschnitt wird dabei ein Prozess vorgestellt, in dem ein *AUV* durch Distanzmessungen zu drei *Überwasser-Bojen* seine Position mittels Multilateration bestimmt (siehe Abbildung 5.2). Grundlage für diese Berechnung lieferte die persönliche Konzept-Mitteilung vom Betreuer im November 2016.

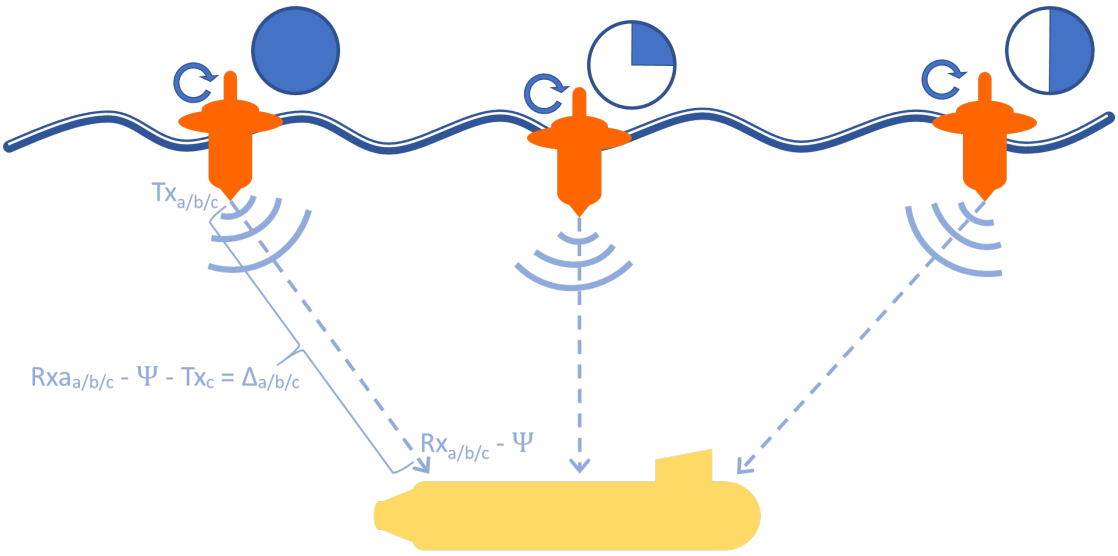


Abbildung 5.2: Positionsbestimmung durch drei Distanzen

Die Bojen befinden sich dabei an der Wasseroberfläche und sind durch direkte Luftverbindung zu Satelliten in der Lage, ihre Position über GPS zu synchronisieren. Das AUV befindet sich an einem ihm unbekannten Ort unter Wasser. Die Bojen senden nun jeweils z.B. einmal stündlich Nachrichten aus, die ihre Position und den Sendezeitpunkt der Nachricht enthalten. Da das Senden von Nachrichten einen höheren Stromaufwand benötigt, als das Empfangen der Nachrichten, soll das AUV keine Nachrichten senden, um aufgrund der begrenzten Akkukapazität Strom zu sparen. *Boje 1* sendet in diesem Beispiel zu jeder vollen Stunde hintereinander drei Nachrichten aus. *Boje 2* sendet analog dazu 15 Minuten nach jeder vollen Stunde und *Boje 3* 30 Minuten nach jeder vollen Stunde aus. Jede Boje sendet mittels Flooding drei Mal. Wir behandeln nun für die Synchronisation eine Einzelboje, für die Multilateration sind alle Bojen zu betrachten. Die drei Aussendezeitpunkte jeder einzelnen Boje seien $\mathrm{Tx_a}$, $\mathrm{Tx_b}$ und $\mathrm{Tx_c}$. Die Aussendungen

beinhalten als Kommunikatonsnachricht den Aussendezeitpunkt. Es wird davon ausgegangen, dass der Flodding-Jitter in diesen versendeten Zeitstempel eingerechnet wurde. Das AUV empfängt die Nachrichten zu den gedrifteten Systemzeiten $\mathrm{Rx_a}$, $\mathrm{Rx_b}$ und $\mathrm{Rx_c}$. Der tatsächliche Empfangszeitpunkt ergibt sich also aus $Rx_{a,b,c} - \psi$, wobei ψ der Offset des AUVs zur tatsächlichen Zeit ist. Die jeweiligen Übertragungszeiten bezeichnen wir als Δ_a, Δ_b und Δ_c. Nun nehmen wir an, dass sich die Übertragungszeiten folgendermaßen zueinander verhalten: $\frac{\Delta_b}{\Delta_a} = \frac{\Delta_c}{\Delta_b}$. Dies lässt sich umstellen zu $\Delta_b^2 = \Delta_a * \Delta_c$. Da die Übertragungszeiten jeweils die Empfangszeit subtrahiert mit der Aussendezeit ist, also $\Delta_{a,b,c} = Rx_{a,b,c} - \psi - Tx_{a,b,c}$, können wir den Offset ψ insgesamt abschätzen zu

$$\psi = \frac{(Rx_b - Tx_b)^2 - (Rx_c - Tx_c)(Rx_a - Txa)}{2Rx_b - 2Tx_b - Rx_c + Tx_c - Rx_a + Tx_a}$$

Dadurch ist die genaue Übertragungszeit bekannt und durch die gemittelte Schallgeschwindigkeit können nun die Distanzen zu den drei Bojen berechnet und anschließend die Position mittels Multilateration bestimmt werden. Dies funktioniert ähnlich wie in handelsüblichen Navigationsgeräten.

Beispiel

Es seien die folgenden Sendezeiten einer einzelnen Boje gegeben:

$$Tx_a = 0$$
$$Tx_b = 5$$
$$Tx_c = 10$$

Die Empfangszeiten seien gegeben mit:

$$Rx_a = 2.5$$
$$Rx_b = 7.6$$
$$Rx_c = 12.71$$

Die Übertragungszeiten seien:

$$delta_a = 1$$
$$delta_b = 1.1$$
$$delta_c = 1.21$$

Somit sollte sich also folgender Offset ergeben:

$$\psi = 1.5$$

Durch Einsetzen in die vorgestellte Formel erhalten wir

$$\begin{aligned}
\psi &= \frac{(Rx_b - Tx_b)^2 - (Rx_c - Tx_c)(Rx_a - Txa)}{2Rx_b - 2Tx_b - Rx_c + Tx_c - Rx_a + Tx_a} \\
&= \frac{(7.6 - 5)^2 - (12.71 - 10)(2.5 - 0)}{2 * 7.6 - 2 * 5 - 12.71 + 10 - 2.5 + 0} \\
&= \frac{6.76 - 2.71 * 2.5}{15.2 - 10 - 12.71 + 10 - 2.5 + 0} \\
&= \frac{-0.015}{-0.01} \\
&= 1.5
\end{aligned}$$

Und dies entspricht genau dem angenommenem Offset. Durch die Annahme einer Schallgeschwindigkeit $c = 1500\,\frac{m}{s}$ ergeben sich somit folgende geschätzte Entfernungen:

$$s_a = 1500m$$

$$s_b = 1650m$$

$$s_c = 1815m$$

Analog erhält man die Entfernungen der beiden für einen Multilaterationsschritt zur Bestimmung der Position wichtigen anderen Bojen. Hierzu sei auf die Masterarbeit „GPS in underwater communication networks" von Dubrovinskaya Elizaveta verwiesen.

5.3 Komplikationen bei der Distanzmessung

In den Prozessformulierungen im vorangegangenen Kapitel 5.2 haben wir nur den Prozess an sich betrachtet und sind davon ausgegangen, dass alle gesendeten Nachrichten auch ankommen. Dies spiegelt allerdings nicht die realen Gegebenheiten wider, denn es kann zu Fehlertypen verschiedenster Art bei der Übertragung kommen.
Einige potenzielle Fehler sollen hier nun zunächst genannt werden, um sie in Kapitel 5.6 in Hinblick auf die Fehlertoleranz näher beleuchten zu können.

Fehlerhafte Checksumme Jede Nachricht enthält am Ende eine von dem Absender aus den eigentlichen Daten berechnete Checksumme. Der Empfänger berechnet ebenfalls die Checksumme aus den Daten und vergleicht diese mit der vom Absender übertragenen Checksumme. Stimmen die Checksummen nicht überein, weist dies in erster Linie auf die Existenz eines Übertragungsfehlers, allerdings nicht über seinen Ausmaß, hin. Beispielsweise kann auch mit den benötigten Informationen alles in Ordnung sein und nur die Checksumme falsch berechnet oder übertragen worden sein. Es sind für die Berechnung die jeweils letzten drei korrekt eingegangenen Nachrichten mit Zeitstempel zu nutzen. Es ist also ein gleitendes Fenster für jeden Ein-hop-Nachbar vorzusehen.

Nachricht kommt nicht an Es kann natürlich auch vorkommen, dass ein Subjekt eine Anfrage sendet und von seinem Kommunikationspartner gar keine Antwort erhält. Dies ist ein sehr spezieller Fall, da dies sehr viele Ursachen haben kann und nicht einmal eine Verbindung als Kommunikationsbasis besteht, wodurch Fehlerursachen eventuell eigenständig durch „Rückfragen“ geklärt werden könnten.

Checksumme stimmt, aber Nachrichteninhalt unstimmig oder unsinnig Wenn nach Übereinstimmungsprüfung der Checksummen auf den ersten Blick kein Fehler vorliegt, können die Informationen dennoch fehlerhaft sein. So kann zum Beispiel der Absender die Daten falsch berechnet haben oder aus anderen Gründen nicht stimmige oder falsche Daten gesendet haben. In diesem Fall liegt der Fehler nicht in der Übertragung, sondern bei der Ermittlung und Absendung der erforderlichen Daten beim Absender. Dieser Fall muss ebenfalls berücksichtigt werden.

5.4 Nachrichten

In diesem Abschnitt sollen die Nachrichtentypen für die Kommunikation der Subjekte zur Distanzmessung definiert werden. Auch hierbei unterscheiden wir zwischen den beiden betrachteten Prozessen, da bei der Distanzmessung zwischen zwei Subjekten weniger Informationen zur Berechnung benötigt werden. Da hier relative Zeiten genutzt werden, muss zum Beispiel kein Zeitstempel übertragen werden.

5.4.1 Distanzmessung zwischen zwei Subjekten

Bei der Distanzmessung mittels *mark-snap* gibt es zwei verschiedene Nachrichten, die sich allerdings vom Nachrichtentyp ähneln: die *mark*-Nachricht und die *snap*-Nachricht. Um die Distanz dieser Vorgehensweise entsprechend ausrechnen zu können, benötigen wir in der Nachricht folgende Informationen:

- **Die Kommunikations-ID**
  Die Kommunikations-ID gibt Aufschluss über den Nachrichtenverlauf, dem diese Nachricht zuzuordnen ist.
- **Den Nachrichten-Absender**
  Der Absender der Nachricht - einer der beiden Teilnehmer am Distanzmessungs-Prozess.
- **Den Nachrichten-Empfänger**
  Der Empfänger der Nachricht - einer der beiden Teilnehmer am Distanzmessungs-Prozess.
- **Die Bearbeitungszeit**
  Die Reaktionszeit, die benötigt wird, um auf die Nachricht zu antworten (also die Zeitspanne zwischen Empfang und Senden).

Wir benutzen für diese Nachrichten das *JANUS*-Protokoll (siehe Kapitel 3.2). Dazu definieren wir den JANUS-Payload mit Task-Zustand (Mark, Snap) [1 bit], eigener ID [4 bit], Processing-Time [8 bit], gemessene eigene Tiefe (optional) [5 bit], gemessene Schallgeschwindigkelt [8 bit] sowie den CRC-Prüfwert der vorangegangen Nachricht [8 bit].

Für die erste mark-Nachricht wird nur der Empfänger, mit dem die Distanz gemessen werden soll, benötigt. Dass es sich hierbei um eine mark-Nachricht handelt, wird im Application-Type-Feld der Nachricht festgelegt.
Die Antwort *snap* erhält nun die vorangegangene Checksumme der 1. mark-Nachricht und

die Bearbeitungszeit in Millisekunden. In der Checksumme der 1. mark-Nachricht ist die Kommunikations-ID und der Nachrichten-Absender und -Empfänger implizit enthalten. Die 2. mark-Nachricht ist gleich der snap-Nachricht aufgebaut.

5.4.2 Distanzmessung zwischen mehr als zwei Subjekten

Für die Distanzmessung am Beispiel der Positionsbestimmung werden mehrere Informationen benötigt. Wie in Kapitel 5.2.2 beschrieben, senden die Bojen stündlich dreimal Nachrichten aus. Diese Nachrichten sind die sogenannten *„Here I am“*-GUWAL-Nachrichten und müssen die Position der Boje und einen Zeitstempel des Sendezeitpunkts der Nachricht enthalten. Der Nachrichtentyp wird in Kapitel 3.4 näher erläutert.

5.5 Modellierung in S-BPM

Wie bereits in Kapitel 5.2 ausführlich beschrieben, sollen zwei verschiedene Szenarien der Distanzmessung betrachtet werden. Einerseits die Distanzmessung zwischen genau zwei Subjekten mithilfe der Methode *mark-snap* und andererseits die Distanzmessung zwischen mehreren Subjekten, in diesem Beispiel drei Subjekten, als Fundament für die Positionsbestimmung mittels Multilateration. Die hier entworfenen Modelle sollen als Grundlage für die Fehlertoleranz-Untersuchung in Kapitel 5.6 dienen.

5.5.1 Distanzmessung zwischen zwei Subjekten

In diesem Abschnitt wird die Methode *mark-snap* in S-BPM modelliert. Zunächst wird die Kommunikationssicht modelliert und erklärt, anschließend werden die internen Subjektansichten der Subjekte modelliert und erklärt. Die auftretenden Subjekte sind in diesem Beispiel ein AUV und eine Boje, die ihre Distanz zueinander bestimmen wollen.

Kommunikationsansicht

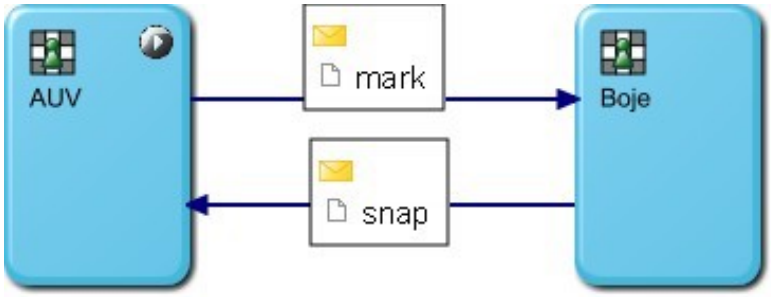


Abbildung 5.3: *mark-snap* - Kommunikationssicht

Abbildung 5.3 zeigt die Kommunikationssicht der *mark-snap*-Methode. Sie besteht aus den zwei Subjekten *AUV* und *Boje*. Die Subjekte tauschen Nachrichten aus, und zwar die *mark*-Nachricht und die *snap*-Nachricht. Die *mark*-Nachricht wird dabei vom AUV an die Boje gesendet und die *snap*-Nachricht von der Boje an das AUV. Das interne Verhalten der beiden Subjekte wird nun im nächsten Abschnitt erläutert.

interne Subjektansicht: AUV

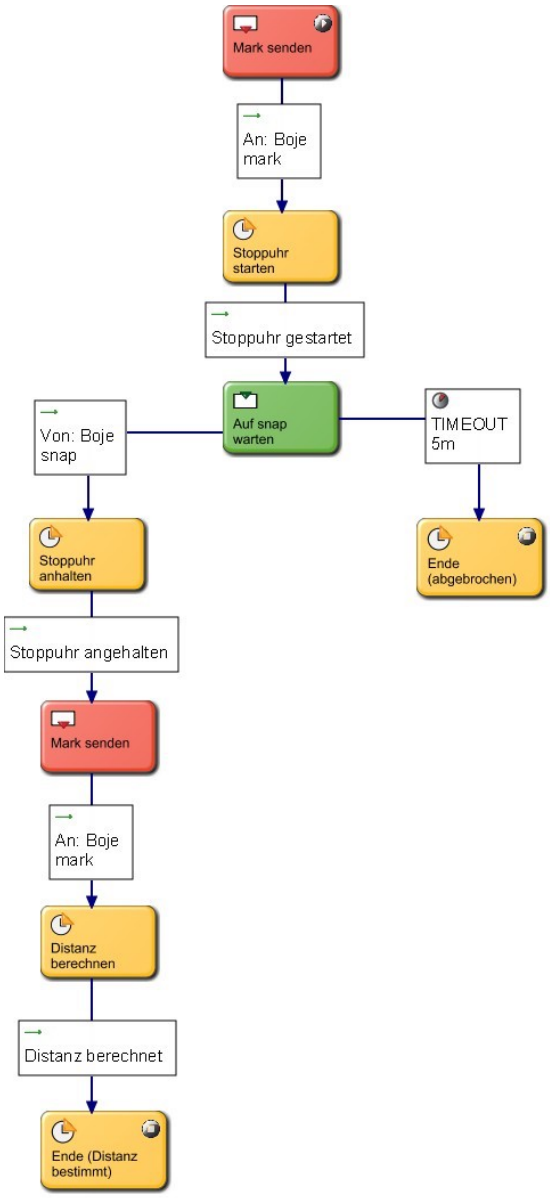


Abbildung 5.4: *mark-snap* - interne Subjektansicht: AUV

Abbildung 5.4 stellt das interne Verhalten des AUVs dar. Der Prozess beginnt insgesamt damit, dass das AUV *mark* an die Boje sendet. Quasi im selben Augenblick speichert es die aktuelle Uhrzeit und startet damit implizit eine Stoppuhr. Dies ist notwendig, damit später aus der Zeitdifferenz und damit der Übertragungsdauer der Nachricht die

Distanz berechnet werden kann (siehe Kapitel 5.2.1). Anschließend wird auf eine Antwort in Form eines *snap* gewartet. Trifft diese innerhalb von 5 Minuten nicht ein, kommt es zu einem Timeout-Fehler und die Distanzmessung ist in diesem Vorgang fehlgeschlagen. Wenn die Antwort-Nachricht innerhalb von 5 Minuten ankommt, wird wieder die Uhrzeit gespeichert und somit implizit die Stoppuhr gestoppt. Im selben Moment wird erneut ein *mark* an die Boje gesendet, damit diese auch die Distanz berechnen kann. Nun kann aus der gestoppten Zeitdifferenz die Distanz berechnet werden und die Distanzmessung war erfolgreich.

interne Subjektansicht: Boje

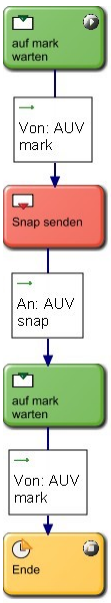


Abbildung 5.5: *mark-snap* - interne Subjektansicht: Boje

Abbildung 5.5 stellt das interne Verhalten der Boje dar. Der Prozess für die Boje verläuft ähnlich wie der des AUVs und beginnt damit, dass auf das *mark* vom AUV gewartet wird. Trifft die Nachricht ein, wird direkt ein *snap* gesendet und auf das zweite *mark* gewartet. Ist dieses eingetroffen, so ist der Prozess beendet. Dieser Prozess kann nun auch so modifiziert werden, dass auch hier die Distanz berechnet werden kann, indem ebenfalls die Zeit gestoppt wird. Davon wurde hier zugunsten der Übersichtlichkeit abgesehen.

Insgesamt können damit beide unsynchronisierten Subjekte die Distanz zueinander auf Basis einer angenommen Schallgeschwindigkeit schätzen.

5.5.2 Distanzmessung zwischen mehr als zwei Subjekten

In diesem Abschnitt wird die Distanzbestimmung von einem AUV zu drei Bojen in S-BPM modelliert (nach dem Schema in Abbildung 5.2), wodurch dann die Position mittels Multilateration (siehe multilateration.com) bestimmt werden kann. Zunächst wird ebenfalls die Kommunikationssicht modelliert und erklärt, anschließend werden wieder die internen Subjektansichten der Subjekte modelliert und erklärt.

Kommunikationsansicht

Abbildung 5.6: Positionsbestimmung - Kommunikationssicht

In Abbildung 5.6 ist die Kommunikationssicht des Prozesses dargestellt. Darin enthalten sind wiederum zwei Subjekte: ebenfalls ein AUV und als Kommunikationspartner Bojen in einer Sonderform des Subjekts, das sogenannte *Multisubjekt*. Ein Multisubjekt beschreibt die Tatsache, dass die Kommunikation mit verschiedenen Instanzen des Subjekts stattfinden kann. Es gibt also nicht genau ein Subjekt, sondern mehrere Ausführungen. Dabei senden lediglich die Bojen *„Here I am“*-GUWAL-Parcel als Nachrichten aus. Wie in Kapitel 5.2.2 beschrieben, soll das AUV aufgrund von Akkusparmaßnahmen selbst keine Nachrichten aussenden.

interne Subjektansicht: Boje

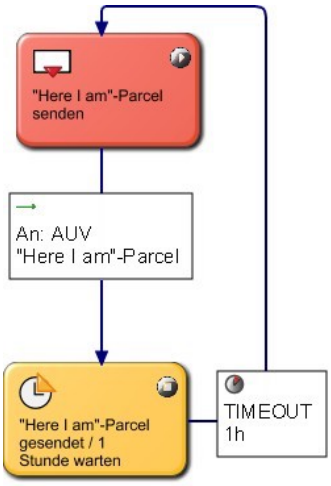


Abbildung 5.7: Positionsbestimmung - interne Subjektansicht: Boje

Abbildung 5.7 stellt das interne Verhalten der Bojen dar. Die Bojen senden zum Beispiel stündliche *„Here I am“*-Parcel an das AUV aus (in diesem Beispiel drei Stück, siehe Kapitel 5.2.2). Die Pakete enthalten unter anderem ihre Position und den Sendezeitpunkt. Nachdem die Pakete gesendet wurden, versetzt die Boje den Prozess für eine Stunde in den Wartezustand, bis sie die Pakete erneut sendet. Die Sendezeitpunkte der Bojen sind versetzt, sodass jede der drei Bojen zu einer unterschiedlichen Zeitspanne nach jeder vollen Stunde sendet.

interne Subjektansicht: AUV

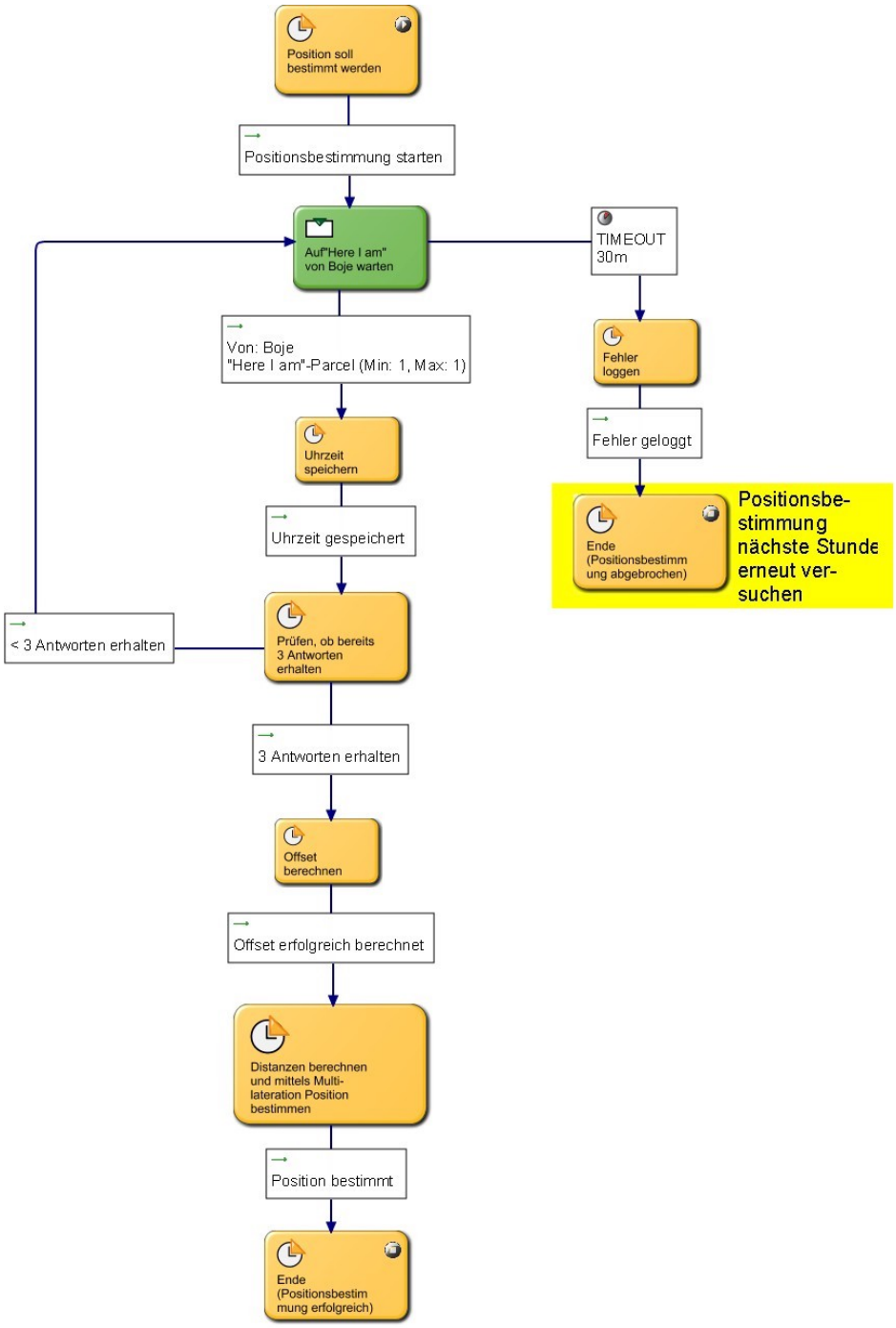


Abbildung 5.8: Positionsbestimmung - interne Subjektansicht: AUV

Abbildung 5.8 stellt das interne Verhalten des AUV dar. Der Prozess startet, wenn eine Positionsbestimmung benötigt wird. Nun wird zunächst auf das erste *„Here I am“*-Parcel gewartet. Nach erhalt eines Paket-Satzes von einer Boje wird die Empfangszeit gespeichert und auf das nächste Paket gewartet, bis insgesamt von drei verschiedenen Bojen empfangen wurde. Danach kann mit der Formel aus Kapitel 5.2.2 der Synchronisations-Offset berechnet, die Distanzen zu den Bojen ermittelt und mittels Multilateration die Positionsbestimmung durchgeführt werden. In diesen Prozess fließen kontinuierlich alle Nachrichtenpakete ein, die einen Sendezeitpunkt beinhalten. Es werden jeweils die letzten drei Aussendungen eines Knotens für die Formel 5.2.2 verwendet.

5.6 Fehlertoleranz

Gerade bei autonomen Vorgängen unter Wasser ist die Toleranz gegenüber Fehlern ein wichtiger Faktor. Fehlertoleranz drückt aus, wie stark Fehler den normalen Prozessfluss beeinträchtigen und klassifiziert damit die Robustheit des Prozesses gegen Fehler. In Kapitel 5.3 wurden einige mögliche Fehlertypen beschrieben. Folgend soll die Toleranz der in Kapitel 5.5 modellierten Prozesse gegenüber diesen Fehlertypen überprüft werden.

Fehlerhafte Checksumme Wenn die Checksumme fehlerhaft ist, ist zumindest eine Nachricht eingetroffen und es besteht eine Verbindung zur Kommunikation. Darauf kann aufgebaut werden und zunächst versucht werden, den entsprechenden Vorgang zu wiederholen. Entweder wird die Anfrage erneut gesendet, oder mit einer entsprechenden Fehlermeldung reagiert und geantwortet. Sollte dies zu keinem besseren Ergebnis führen, könnte zunächst der gleiche Prozess mit anderen Subjekten getestet werden, um Eigenversagen ausschließen zu können. Liegt der Fehler beim Kommunikationspartner, kann intern versucht werden, die betreffende Fehlerstelle der Nachricht ausfindig zu machen. Wenn dies gelungen ist, kann überprüft werden, ob diese Informationen vor den Anwendungsfall der Distanzmessung notwendig ist. Ansonsten kann, wenn die Distanzmessung nicht unbedingt erforderlich ist, zum Beispiel durch vorhandene Ersatzsubjekte mit denen gemessen werden kann, auf eine Distanzmessung verzichtet werden.
Gegen diesen Fehlertyp sind die Prozesse mittelmäßig tolerant. Es kommt darauf an, ob sich die Fehler häufen oder nach der ersten Wiederanfrage bereits gelöst sind. Je weiter sich die Fehlersuche verzweigt, desto intoleranter wird der Prozess gegenüber dem Fehler.

Nachricht kommt nicht an Sollte die erwartete Nachricht gar nicht erst ankommen, kann außer erneuter Sendeversuche zunächst nicht viel getan werden. Möglicherweise ist der Kommunikationspartner nicht in Reichweite, nach kleineren Positionsänderungen könnte der Vorgang erneut gestartet werden. Sollte auch dies keine Abhilfe des Problems verschaffen, muss der Prozess abgebrochen und zu einem späteren Zeitpunkt in einem neuen Versuch erneut ausgeführt werden.
Gegen diesen Fehlertyp ist das System am wenigsten tolerant, da eine Kommunikation mit dem anderen Subjekt für eine Distanzmessung zwingend erforderlich ist.

Checksumme stimmt, aber Nachrichteninhalt unstimmig oder unsinnig Durch interne Prüfalgorithmen kann getestet werden, ob die bestimmten Distanzen sinnig sind oder so gar nicht auftreten können (zum Beispiel wenn die ungefähre Distanz bekannt ist oder

beim Auftreten viel zu hoher oder negativer Werte, auch die beim genutzten Frequenzbereich geschätzte maximal mögliche Entfernung). Dann liegt wahrscheinlich ein einmaliger Messfehler vor, der beim nächsten Messversuch nicht mehr auftreten sollte. Tritt dieser Fehler allerdings regelmäßig auf, ist dies am wahrscheinlichsten auf mangelhafte Hard- oder Software der beteiligten Subjekte zurückzuführen (zum Beispiel Berechnungsfehler, fehlerhafte Zeitsysteme oder absichtliche Fakenews).
Da dieser Fehler in der Regel allerdings wahrscheinlich nur vereinzelt auftritt und Distanzmessungen in regelmäßigen Abständen durchgeführt werden, ist das System hiergegen relativ tolerant.

Insgesamt können bei der Distanzmessung zwischen mehreren Objekten immer jene letzten drei Nachrichten gewählt werden, bei denen Sende- und Empfangszeitpunkt bekannt sind. Somit können die Distanzen grob geschätzt werden.

5.7 Fazit und Ausblick

Das Kapitel Distance Measurement sollte einen Überblick über die Schwierigkeiten der Distanzmessung unter Wasser und deren enormen Unterschied zur Distanzmessung in der Luft geben. Es wurden einführend die Prozessabläufe besprochen, um darauf aufbauend mögliche Schwierigkeiten anzusprechen. Weiterhin wurden die Prozesse in S-BPM modelliert und abschließend wurde auf die Fehlertoleranz der Prozesse eingegangen.
Die beiden Prozesse sind dabei sowohl in der Durchführung, als auch in der Fehleranalyse nur grob beschrieben worden. Es gibt durchaus deutlich mehr Fehlerquellen, die auftreten können, welche für eine genauere Betrachtung auch mit modelliert werden können. Aber auch die Prozesse an sich wurden nur grundlegend aufgestellt. Die Positionsbestimmung wird natürlich präziser, wenn die Distanz zu mehreren Bojen gemessen werden würde und diese zur genaueren Uhrzeit-Synchronisation mehrmals in kürzeren Zeitintervallen aussenden würden. Der Prozess kann beliebig komplex ausgebaut werden und dient nur als einfache Grundlage unter Beachtung lediglich einfacher Annahmen (zum Beispiel konstanter Schallgeschwindigkeit). Daher dienen die Prozesse nur als Basis und können unter einem gewähltem Fokus präzisiert werden.

6 Postman

Thies Petersen

6.1 Einleitung

Dieser Teil des Projektes beschäftigt sich mit der Frage, wie Nachrichten von unter dem Wasser weit verteilten Sensorknoten abgeholt und an diese übermittelt werden können. Erschwert wird diese Aufgabe dadurch, dass die exakten Positionen der Knoten unbekannt sind und dass unter Wasser eine Navigation mittels GPS nicht möglich ist.

6.1.1 Ein Gleiter als Postbote

Eine mögliche Lösung ist, einen Unterwassergleiter als Postboten für die Sensorknoten einzusetzen. Anders als bei herkömmlichen Unterwasserfahrzeugen, ist dieser Gleiter in der Lage, ohne Propellerantrieb auszukommen. Ausgenutzt wird dabei die Möglichkeit, die Dichte im Druckkörper des Gleiters zu ändern. Dabei wird, ähnlich wie beim von U-Booten bekannten Verfahren, Wasser herein- oder herausgepumpt. Dies bewirkt, dass der Gleiter absinkt oder aufsteigt. Vorstellen kann man sich diese Bewegung wie die Zacken eines Sägeblattes. Der Energieaufwand verringert sich dadurch enorm, denn der Gleiter benötigt lediglich Energie für den Pumpvorgang. Damit ist ein langer Einsatz möglich, bevor ein Nachladen nötig wird. Die Steuerung kann zum Beispiel durch Verlagerung des Gewichts innerhalb des Gleiters erfolgen. Die Navigation des Gleiters erfolgt unter Wasser mittels Kompass. Nach einem erneuten auftauchen können die Navigationsdaten mit GPS abgeglichen und eventuell angepasst werden. Ohne eigenen Antrieb, der zu energieintensiv wäre, hängt die Geschwindigkeit von der vorhandenen Meeresströmung ab und beträgt in der Regel weniger als 1 km/h.

6.1.2 Einsatz

Verfahren mit Sensorknoten und Gleiter werden unter anderem bei der Überwachung von Entwicklungen in der Tiefsee oder unter dem ewigen Eis eingesetzt, in Gebieten also, in

denen es nicht ohne weiteres möglich ist, die ermittelten Daten der Knoten einfach abzufragen. Messergebnisse können zum Beispiel die Beobachtung und Überwachung von Temperaturen und Meeresströmungen sein. Fest installierte Knoten sammeln die Daten und speichern sie. Der Gleiter kommt periodisch vorbei und holt die gesammelten Informationen ab. Es ist aber auch der Einsatz des im Weiteren geschilderten Verfahrens in der Wartung von Tiefsee-Infrastruktur vorstellbar: beispielhaft können autonome Unterwasserfahrzeuge Transkontinentalkabelverbindungen abfahren und auf Beschädigungen oder Störungen untersuchen. So können Reparatureinheiten im Bedarfsfall zielgerichtet eingesetzt werden.

6.1.3 Ablauf

Im Einsatz sinkt der Gleiter auf die vorgegebene Tiefe ab. Dort angekommen, sendet er die Nachricht seiner Ankunft an die von seiner aktuellen Position erreichbaren Knoten. Sofern ein Knoten diese Nachricht empfängt, sendet dieser eine Antwort mit seinem Namen, seiner MAC-Adresse, Zeit, wie viele Päckchen er senden wird und die Checksumme der Päckchen sowie einen Prioritätswert der Nachrichten. Sollte diese Antwort binnen einiger weniger Minuten die Einzige sein, wird mit dem Senden der Päckchen begonnen. Gehen weitere Antworten von anderen Knoten ein, wird die Verzögerung erhöht um zu verhindern, dass mehrere Knoten gleichzeitig senden und Kollisionen der Pakete entstehen. Zusätzlich wird ein, bei Datenübertragungen eigentlich unerwünschter Jitter, also eine deterministische Verzögerung, hier dazu eingesetzt, um Kollisionen von Datenpaketen zu vermeiden.

Das Nachrichtenprotokoll des Postboten wiederholt eingegangene Pakete. So können die Absenderknoten überprüfen, ob ihre Nachrichten korrekt eingegangen sind und diese abhaken oder erneut senden. Sind alle Pakete von den Knoten an den Postboten übertragen, sendet dieser die eventuell mitgebrachten Nachrichten an die Knoten. Dieser Vorgang wiederholt sich, so lange sich Knoten beim Gleiter melden. Kommen keine weiteren Anmeldungen von Knoten, steigt der Gleiter nach einer festgelegten Zeit wieder auf Meereshöhe, sendet die empfangenen Pakete ab, überprüft seine Position und sinkt dann wieder auf die vorgegebene Tiefe.

6.1.4 Besonderheit der Kommunikation unter Wasser

Die Kommunikation unter Wasser stellt im oben geschilderten Vorgehen eine besondere Herausforderung dar. Einmal ist vorab die Route des Postboten und die Position von

Sender und Empfänger nicht bekannt. Außerdem ist die Übermittlung störanfällig und nicht sichergestellt, dass sie unterbrechungsfrei stattfindet. Es wird also eine Technik benötigt, die Unterbrechungen in der Übertragung toleriert. Bei diesem Delay Tolerant Networking (kurz: DTN) werden Nachrichten in einzelne, kleinere Päckchen zerlegt. Diese Bündel von Päckchen werden von allen aktuell verfügbaren Knoten nach dem store-and-forward Prinzip übertragen (Siehe Kapitel 3). Das heißt, Nachrichten werden so lange von allen Knoten weitergeleitet, bis der Empfänger durch das eigene Weiterleiten bestätigt, dass er das jeweilige Paket erhalten hat. Empfangene Päckchen können abgehakt werden. Diese Übertragungstechnik ist zwar deutlich langsamer als bekannte Routingverfahren, im Gegensatz zu diesen ist in einer lebensfeindlichen Umwelt aber sichergestellt, dass die Daten den Empfänger erreichen.

6.1.5 Ausblick

Im weiteren Verlauf der Arbeit wird der oben geschilderte Prozess der Nachrichtenübermittlung mittels Postboten modelliert. Ein besonderer Schwerpunkt liegt dabei in der Betrachtung möglicher Fehler und dem Umgang mit diesen, im Hinblick auf die Fehlertoleranz. Fehlertoleranz bedeutet in diesem Zusammenhang, wie erreicht werden kann, das Fehler erkannt, beziehungsweise wie Nachrichten trotz Fehlern dennoch ausgetauscht und verwertet werden können

6.2 Erste Prozessformulierung

Vor der im weiteren Verlauf dargestellten konkreten Modellierung, ist es Ziel dieses Abschnittes, den Prozess des Postman und der Knoten textuell in natürlicher Sprache darzustellen.

6.2.1 Prozesssubjekte

Es wird in diesem Abschnitt der Arbeit das Verhalten von zwei Subjekttypen betrachtet.

Autonomous Underwater Vehicle (AUV): Unter AUVs verstehen wir im weiteren Verlauf dieser Ausführungen die unter Wasser eingesetzten, verteilten Knoten. Es können 1 bis n verschiedene Knoten eingesetzt werden. In der Praxis sind es aus Kostengründen möglichst wenige. Die Knoten können fest installiert oder auch beweglich sein. Für den Prozess ist es unerheblich, so lange die Knoten das vorher definierte Einsatzgebiet nicht verlassen.

Postman: Der Postman ist, wie bereits ausgeführt, für das Einsammeln und Austragen von Nachrichten zuständig. Er erhält von einer Basis die an die AUVs gerichteten Nachrichten. Von den AUVs sammelt er die Nachrichten an die Basis ein und liefert diese nach dem Auftauchen bei dieser ab.

6.2.2 Prozesselemente und Ablauf

Nachrichten einsammeln Im Anfangszustand erreicht ein Postbote ein Gebiet und sendet periodisch ein Signal, dass seine Ankunft ankündigt ("Postman is here"). Dies geschieht so lange, bis eine Antwort von einem Knoten eingeht. Empfängt ein Knoten dieses Signal, sendet er eine vereinbarte Antwort, das sogenannte "Postman-Reply", welches seinen Namen, seine MAC-Adresse, Systemzeit, die Anzahl an Päckchen, die er senden wird, und die Checksumme der Päckchen sowie einen Prioritätswert der Nachrichten enthält. Der Postman registriert die Antwort und weiß nun, dass es dieses AUV gibt und ob dieses Nachrichten übermitteln möchte. Näheres zur Kontaktaufnahme zwischen Knoten (hier Postman und AUV) wird im Kapitel 4 betrachtet.

Um Kollisionen mit den eventuellen Nachrichten anderer AUV zu verhindern, wartet der Postman ab, ob sich noch weitere anmelden.

Daraus resultieren zwei Fälle: Entweder es hat sich nur ein AUV gemeldet, oder es gibt

mehrere, die mit dem Postman interagieren wollen.

Im ersten Fall gibt der Postman dem AUV das Signal, dass seine Nachrichten nun übermittelt werden können. Das AUV sendet die Nachrichten. Anschließend wiederholt der Postman die empfangenen Nachrichten mittels Flooding und gibt so dem AUV die Möglichkeit, die richtig übertragenen Nachrichten abzuhaken. Ist eine Nachricht fehlerhaft, hat dieser die Möglichkeit, seine Nachricht erneut zu senden (solange der Postman noch in Reichweite ist, siehe Unterabschnitt Fehlertoleranz). Nachdem alle Nachrichten vom AUV übermittelt wurden, wechselt dieser wieder in den Empfangsmodus und gibt dem Postman so die Möglichkeit, seine eventuell für diesen Empfänger mitgebrachten Nachrichten zu übertragen

Der zweite Fall ist umfangreicher. Hier haben sich mindestens zwei AUV beim Postman angemeldet. Dieser muss nun vermeiden, dass die AUV ihre Nachrichten durcheinander senden und so eventuell keine der Nachrichten verständlich übertragen wird. Dies gelingt, in dem sowohl der Postman als auch das Nachrichtenprotokoll eine Verzögerung verwenden. Diese sorgt dafür, dass die Verbindungen nacheinander abgearbeitet werden. Der Umgang mit dem einzelnen AUV erfolgt dann nach dem oben geschilderten Schema. Auch hier werden die übertragenen Nachrichten mittels Flooding wiederholt und können vom jeweiligen AUV abgehakt werden.

Nachrichten abliefern Sofern der Postman Nachrichten hat, übermittelt er diese nach dem Abholen der Nachrichten, das AUV wiederholt sie und gibt so, analog zu dem bereits oben geschilderten Vorgehen, dem Postman die Möglichkeit, zu überprüfen, ob seine Nachrichten korrekt übermittelt wurden. Ist dies der Fall, endet die Interaktion zwischen Postman und diesem AUV. Für die Übertragungen von Nachrichten vom Postman an einen einzelnen konkreten Knoten kann der Postman auf das explizite Acknowledgement des Nachrichtenprotokolls zurückgreifen. Hat der Postman alle Knoten abgearbeitet, kann er zur Basis zurückkehren oder mittels Ping weitere eventuell nun erreichbare Knoten über seine Ankunft informieren. Dieses Prinzip ist GUWMANET implementiert.

6.3 Komplikationen beim Einsatz des Postman

Beim Einsatz des Postmans können verschiedene Komplikationen auftreten. Einige sollen nun benannt und im weiteren genauer betrachtet werden, insbesondere im Hinblick auf die Fehlertoleranz.

Mindestens eine Nachricht wurde fehlerhaft übertragen Die Nachricht weicht in ihrer Checksumme von derjenigen, die der Postman ermittelt hat, ab. Dies sagt erst einmal nichts über die inhaltliche Qualität der Nachricht aus. Eventuell ist nur ein einziges Bit an der Position ungültig. Postman und AUVs sind zu einer qualitativen Analyse nicht in der Lage und können lediglich feststellen, dass die Nachricht nicht wie erwartet übermittelt wurde.

Der angemeldete Knoten ist während der Übertragung nicht mehr in Sende-/ Empfangsreichweite des Postman Insbesondere wenn sich in einem Gebiet mehrere Knoten beim Postman anmelden, ist die Reihenfolge in der sie Nachrichten senden und empfangen dürfen elementar für den erfolgreichen Verlauf der Operation. Wird ein Knoten, der schon am Anfang am äußeren Rand der Route liegt, erst spät an die Reihe genommen, ist die Wahrscheinlichkeit groß, dass dieser zu diesem Zeitpunkt nicht mehr in Reichweite ist oder während der Übertragung außer Reichweite gerät.

Es gab eine fehlerhaft übertragene Nachricht an den Knoten, aber mittlerweile ist der Knoten nicht mehr in Reichweite, erneutes Senden also nicht möglich Der Postman stellt fest, dass der Knoten seine Nachricht nicht korrekt bekommen hat. Er ist aber nicht mehr in direkter Reichweite zum Knoten.

Es meldet sich ein Knoten beim Postman an, der nicht zum Netzwerk gehört Der Postman bekommt die Anmeldung eines Knotens, der nicht auf seiner Liste steht. Dies kann ein Knoten sein, der unabsichtlich in das Gebiet gekommen ist, zu einer anderen Organisation/Messreihe gehört aber auch ein Knoten, der absichtlich von einer fremden Organisation in das Gebiet gesendet wurde.

Ein "fremder"Postman pingt die Knoten an Bei den Knoten meldet sich ein Postman an, dieser trägt allerdings nicht die Kennung des bekannten Postmans. Dies kann dennoch ein Postman sein, der von der eigenen Organisation gesendet wurde, es kann allerdings auch ein Fremder sein, der Informationen abfischen soll.

Ein zweiter Postman ist gleichzeitig in dem Gebiet Während der Postman mit den Knoten interagiert, kommt ein weiterer Postman in das Gebiet.

Diese sechs geschilderten Komplikationen stellen nur einen Bruchteil der möglichen Fälle dar, sollen aber exemplarisch zeigen, mit welchen Problemen bei der Kommunikation

und dem Nachrichtenaustausch im Anwendungsfall Unterwasserkommunikation zu rechnen ist.

6.4 Nachrichten zwischen Postman und AUV

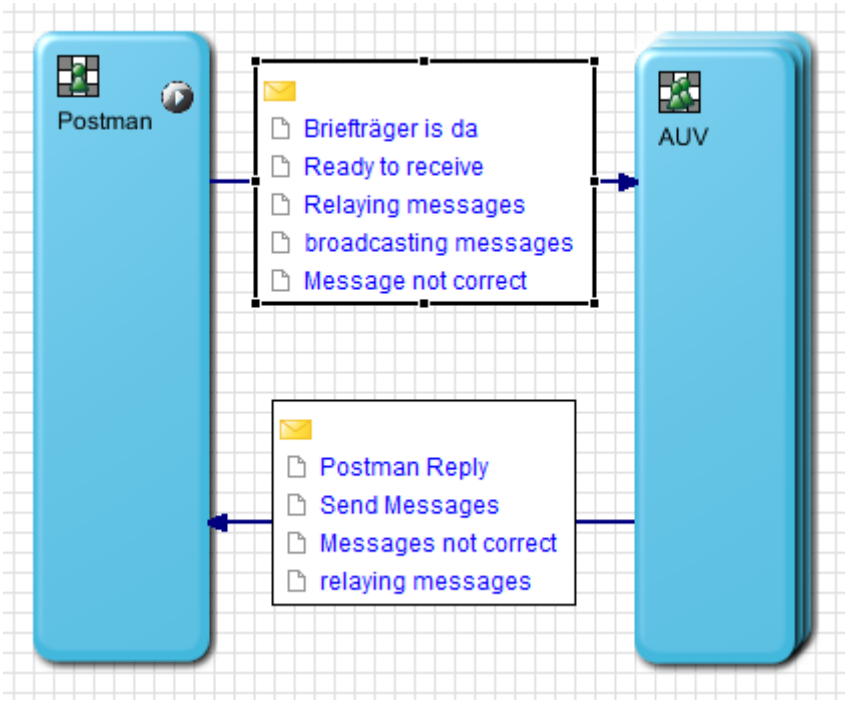


Abbildung 6.1: Zwischen Postman und AUVs ausgetauschte Nachrichten

Wie das verwendete Nachrichtenprotokoll generell Nachrichten zwischen Knoten austauscht, wurde bereits im Kapitel zu Flooding ausgeführt. Um Redundanzen zu vermeiden, wird an dieser Stelle auf eine detaillierte Beschreibung des Nachrichtenaustausches verzichtet. Stattdessen werden die zwischen AUV und Postman ausgetauschten Nachrichtentypen betrachtet. Diese können der obigen S-BPM Modellierung entnommen werden. Nachfolgend eine kurze Beschreibung.

Briefträger ist da Der Postman pingt diese Nachricht periodisch, so lange bis eine Reaktion in Form der Meldung eines AUV erfolgt. Sie enthält neben seinem Namen zur Authentifizierung auch noch die MAC-Adresse und zur Abstandsmessung seine Systemzeit (siehe Kapitel Distance Measurement).

Postman Reply Die AUVs melden mit dieser Nachricht an den Postman zurück, dass sie seine Ankunft registriert haben und bereit zur Interaktion sind. Sie melden ihren Namen, die MAC-Adresse, Systemzeit, den aktuellen Standort, die Anzahl an Päckchen, die sie senden wollen, und deren Checksumme sowie einen Prioritätswert der Nachrichten.

Ready to receive Der Postman sendet diese Nachricht an das AUV, um diesem das Signal für die Übermittlung der Nachrichten zu geben.

Send Messages Das AUV sendet seine Nachrichten. Anders als bei anderen in dieser Arbeit geschilderten Prozessen, ist es hier nicht notwendig, dass die Nachrichten ein speziell für diesen Prozess festgelegtes Schema verwenden, denn sowohl der Postman als auch die AUV verwenden das selbe Nachrichtenprotokoll. Einzige Restriktion ist also, dass die Nachrichten sich an das dort festgelegte Schema halten. Die für die Nachrichtenübermittlung notwendigen Daten, wie Name und Standort, werden bereits beim ersten Kontakt ausgetauscht.

Relaying messages Der Postman wiederholt die empfangenen Nachrichten jeweils, damit das AUV verifizieren kann, dass seine gesendete Nachricht korrekt übermittelt wurde.

Message not correct Das AUV stellt fest, dass eine Nachricht nicht korrekt übertragen wurde und teilt dies dem Postman mit.

Broadcasting Messages Der Postman übermittelt nach Abschluss des Nachrichtenempfanges seine eventuell mitgebrachte Post an das AUV.

Relaying messages Das AUV wiederholt die empfangenen Nachrichten.

Message not correct Der Postman stellt fest, dass eine Nachricht nicht korrekt übertragen wurde, informiert das AUV darüber und sendet die Nachricht erneut.

6.5 Modellierung in S-BPM

6.5.1 S-BPM Prozess des Postman

Es folgen nun die Modellierungen der geschilderten Prozesse in S-BPM, beginnend mit dem Prozess des Postman.

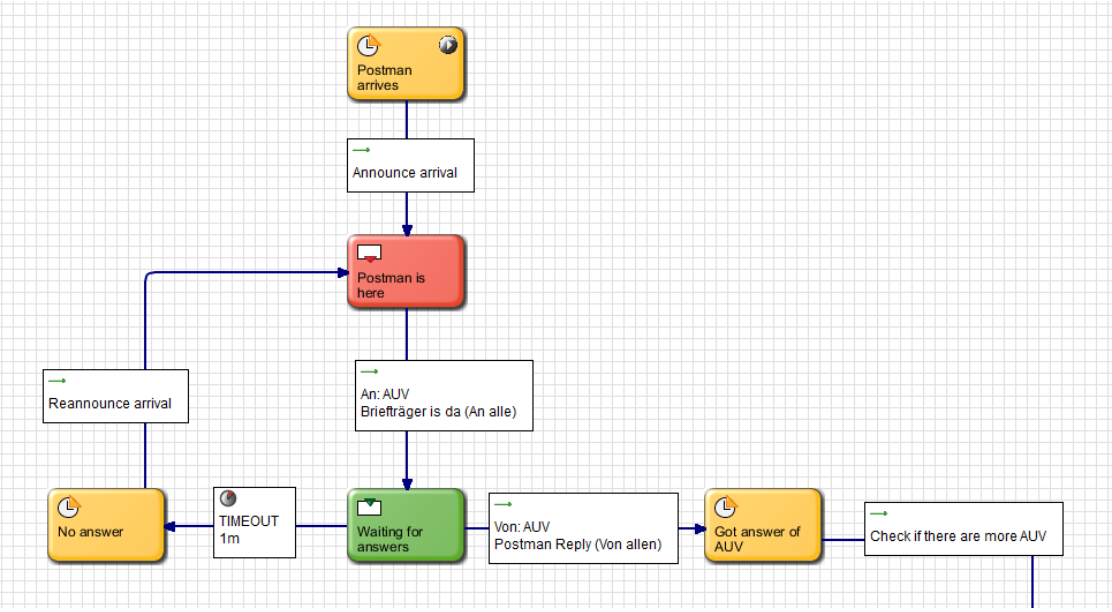


Abbildung 6.2: Kontaktaufnahme zwischen Postman und AUV

In Abbildung 2 ist der geschilderte Beginn des Prozesses dargestellt. Der Postman erreicht ein Gebiet und gibt seine Ankunft bekannt. Er wartet auf eine Reaktion, während er seinen Weg fortsetzt. Nach einem Timeout wiederholt er sein Ankunftssignal. Im Bereich rechts unten ist die erste Reaktion modelliert. Nun wartet der Postman auf eventuell weitere Reaktionen.

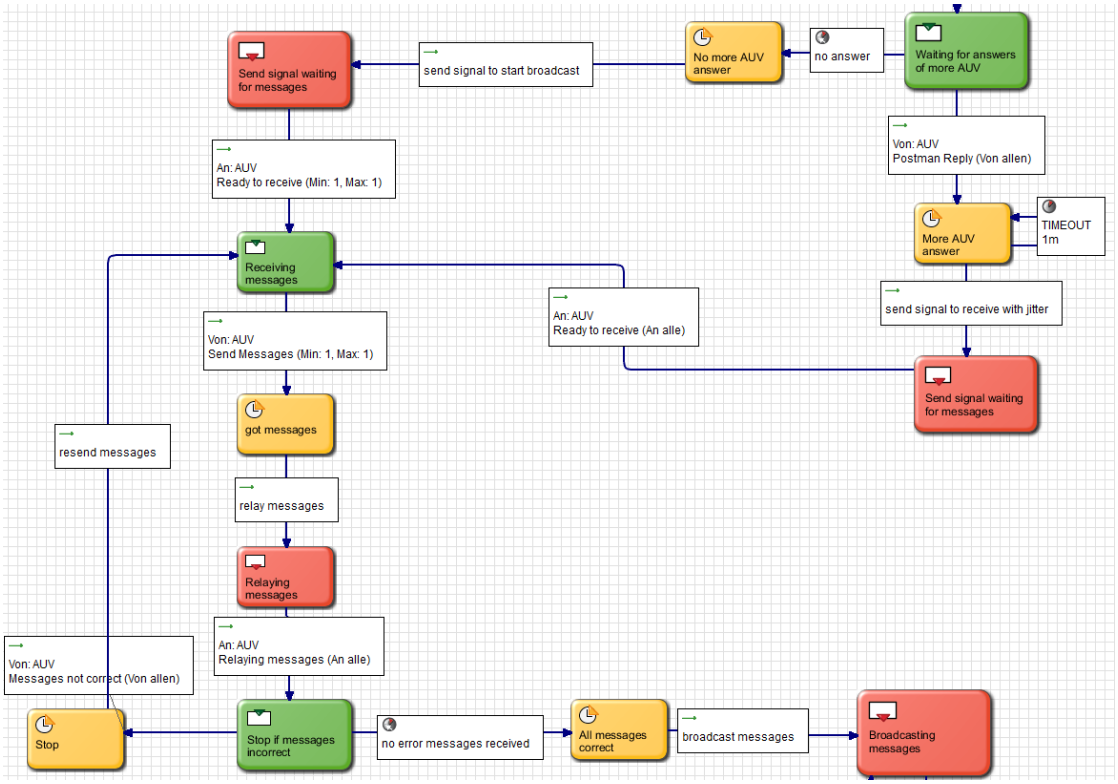


Abbildung 6.3: Nachrichten vom AUV empfangen

In Abbildung 3 wird der Prozess im rechten, oberen Bereich fortgesetzt. Der Postman wartet, nachdem sich ein AUV bereits angemeldet hat, auf weitere Reaktionen. Geht nach einer festgelegten Zeitspanne keine Reaktion ein, sendet er an den angemeldeten AUV das Startsignal zum Übertragen der Nachrichten (links oben). Im Fall, dass eine weitere Anmeldung eingeht, wechselt der Prozess in eine Funktion ("More AUV answer"), in der erneut gewartet wird, ob weitere Anmeldungen kommen. Eingehende Anmeldungen werden einer Liste hinzugefügt Nach einem Timeout ohne weitere Meldungen, wird auch hier das Signal zum Senden der Nachrichten an den ersten AUV auf der Liste gesendet. In beiden Fällen wechselt der Prozess danach in den Empfangsstatus ("Receiving Messages"). Die empfangenen Nachrichten werden wiederholt und die AUVs können die richtig empfangenen Nachrichten abhaken oder den Postman informieren, dass eine Nachricht fehlerhaft ist. Im Fehlerfall wird eine erneute Übertragung angestoßen.
Wurden alle Nachrichten des Knotens korrekt übertragen, wurde also keine Stop-Meldung empfangen, kann der Postman nun seinerseits Nachrichten senden (rechts unten in der Abbildung).

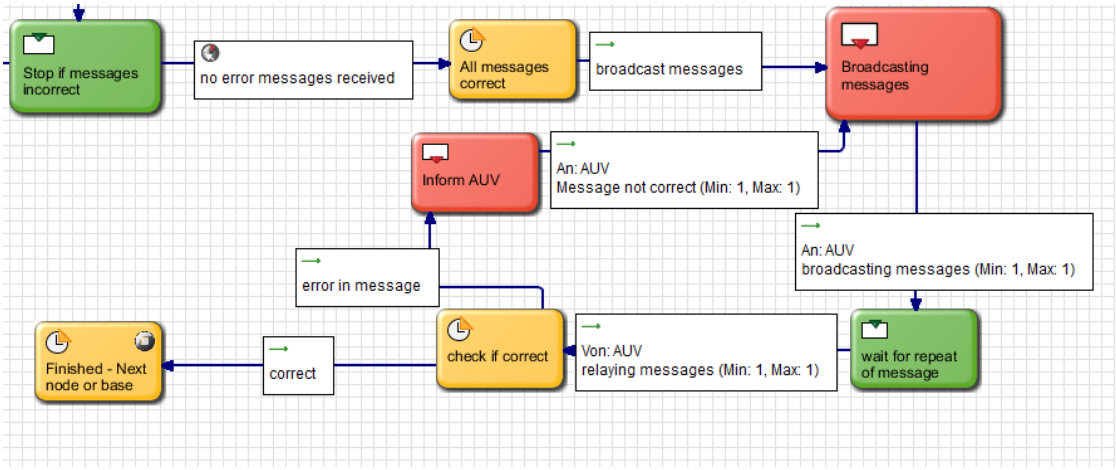


Abbildung 6.4: Nachrichten an AUV senden

Im Prozess des Postman befinden wir uns nun im Bereich rechts oben in Abbildung 4. Der Postman sendet seine Nachrichten an ein konkretes AUV. Dieser wiederholt die eingegangenen Nachrichten, um dem Postman die Möglichkeit zu geben, zu überprüfen, ob seine Nachrichten korrekt übermittelt wurden. Ist dies nicht der Fall, informiert der Postman das AUV, damit dieser erneut in den Empfangszustand wechseln kann und wiederholt danach die fehlerhafte Nachricht. Wurden alle Nachrichten korrekt übermittelt, wechselt der Prozess in den Endzustand und kann entweder mit dem nächsten AUV fortfahren oder zur Basis zurückkehren und die Nachrichten abliefern.

6.5.2 S-BPM Prozess der AUV

Es folgt nun die Modellierung des Prozesses zum Empfangen und Senden der AUVs.

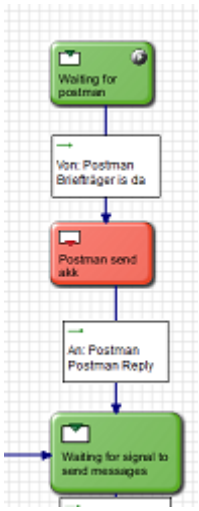


Abbildung 6.5: Kontaktaufnahme mit Postman

Abbildung 5 zeigt den ersten Abschnitt des modellierten Prozesses des AUV. Zu Beginn wartet ein Knoten auf die Ankunft eines Postman. Empfängt er die vereinbarte Nachricht, die die Ankunft ankündigt, sendet er das festgelegte „Postman reply" und wartet auf das Signal des Postmans, dass die Nachrichten übertragen werden können.

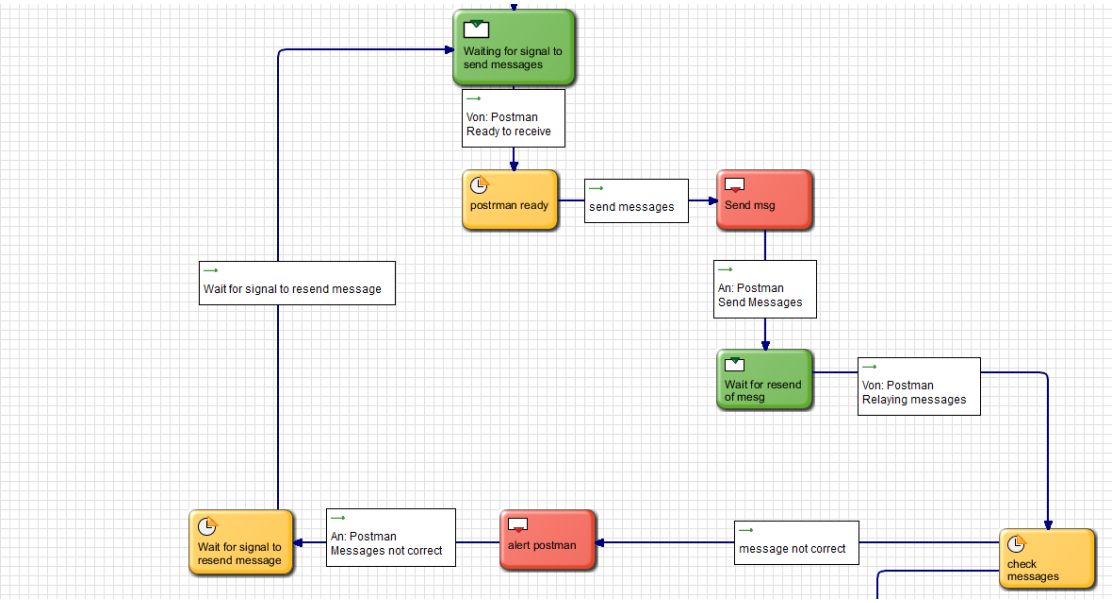


Abbildung 6.6: Nachrichten an Postman senden

Abbildung 6 setzt den Prozess fort. Das AUV erhält vom Postman das Signal, dass seine Nachrichten nun übertragen werden können. Nachdem die Nachrichten übertragen wurden, wartet das AUV auf die Wiederholung und überprüft die Nachrichten auf Korrektheit. Sollte dabei ein Fehler gefunden werden, benachrichtigt er den Postman und wartet auf die Aufforderung, die Nachricht erneut zu senden.

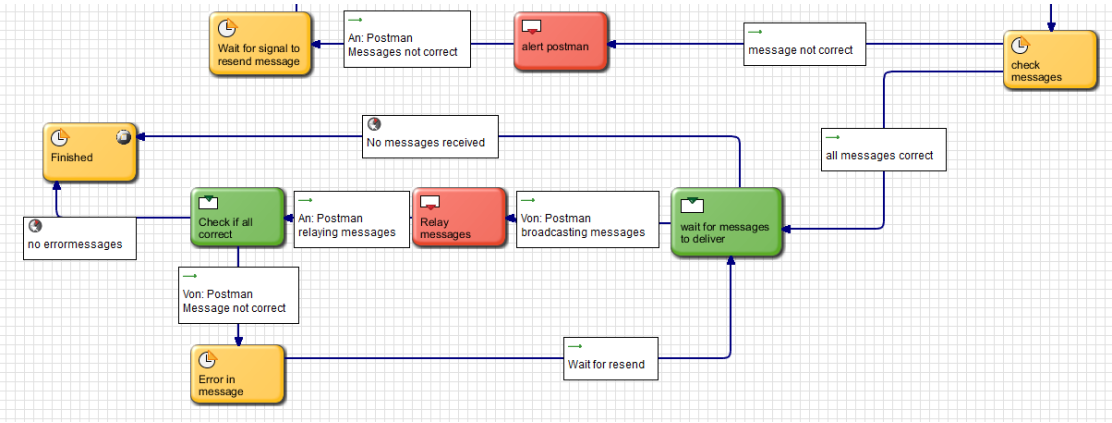


Abbildung 6.7: Nachrichten vom Postman empfangen

Abbildung 7 zeigt nun abschließend, den Teil des Prozesses, der folgt, nachdem alle Nachrichten des AUV korrekt an den Postman übergeben wurden. Das AUV wechselt danach in den Empfangszustand. Sofern er Nachrichten erhält, wiederholt er diese, damit

der Postman überprüfen kann, ob sie korrekt sind. Falls der Postman Fehler feststellt, wartet das AUV auf ein erneutes Senden. Befindet sich der Prozess im Zustand „wait for messages to deliver“ und gehen keine Nachrichten ein, wechselt er nach einer festgelegten Zeitspanne in den Endzustand. In diesen wechselt der Prozess auch, wenn nach dem Wiederholen kein Einwand des Postman verzeichnet wird.

6.6 Fehlertoleranz

Ein wichtiger Aspekt ist der Umgang mit möglichen Fehlern. In Abschnitt 6.3 wurden mögliche Komplikationen bereits skizziert. Hier soll nun der Umgang mit diesen betrachtet werden.

Nachrichten unvollständig/fehlerhaft Zu unterscheiden ist, ob eine Nachricht als ganzes technisch defekt und unbrauchbar ist oder Teile (einzelne Pakete) nicht übertragen wurden. In beiden Fällen wird versucht, ein erneutes Übertragen der Nachricht anzustoßen. Fehlen Teile der Nachricht, weil der Knoten während der Übermittlung z.B. außer Reichweite geraten ist, wird die Nachricht dennoch gespeichert und markiert. Aufgrund des Aufbaus des Netzwerkes, besteht die Möglichkeit, dass ein Knoten in Reichweite des Postmans die fehlenden Pakete noch wiederholt und sie so komplett gespeichert werden kann. Sollte dieser Fall nicht eintreten, wird dennoch immerhin ein Teil der Nachricht aufgenommen und an die Basis übergeben, wo der komplette Sinn oder die relevanten Daten eventuell herausgelesen werden können.

Knoten nicht mehr in Reichweite Mögliche Lösung: Um im Vorfeld zu verhindern, dass ein Knoten während des Senden-/Empfangenvorganges außer Reichweite ist, wird mittels Distance Measurement die relative Entfernung zur eigenen Position ermittelt. Der Postman kann durch die vom Kompass erhaltenen Daten den weiteren Verlauf der Route schätzen. Zusammen mit der ermittelten Position des Knotens ist es möglich, eine Reihenfolge aufzustellen, wann welcher Knoten ungefähr außerhalb der Reichweite sein wird. Anhand dieser Information kann der Postman entscheiden, mit welchem Knoten begonnen werden soll. Zu beachten ist allerdings weiterhin, dass Knoten mit Nachrichten von hoher Priorität bevorzugt behandelt werden müssen.

Nachricht von Postman an Knoten fehlerhaft, aber Knoten nicht mehr in Reichweite Der Postman empfängt die Nachricht, dass seine Nachricht an einen Knoten fehlerhaft empfangen wurde. Im Regelfall würde das Nachrichtenprotokoll ein erneutes Senden der Nachricht anstoßen. Ist der Knoten allerdings mittlerweile außer Reichweite des Postman, ist dies nicht mehr ohne weiteres möglich. Eine Lösung für dieses Problem kann dadurch realisiert werden, dass der Postman seine Nachricht erneut aussendet und diese mit einem expliziten Acknowledge für den Knoten, den die Nachricht fehlerhaft erreicht hat, versieht. Durch das verwendete Flooding besteht die Möglichkeit, dass der Zielknoten zwar nicht mehr in Reichweite sein mag, allerdings zwischen aktueller Position des

Postman und dem Knoten eventuell weitere Knoten liegen. Diese empfangen die Nachricht des Postman und leiten sie inklusive des Acknowledge weiter. So lange immer ein näher zum Zielknoten platzierter Knoten vorhanden ist, wird die Nachricht weitergegeben und kommt am Ende doch noch beim Zielknoten an. Dieser sendet ein Acknowledge an den Postman zurück, welches analog zum geschilderten Vorgehen von den anderen Knoten des Netzwerkes weitergeleitet wird. Sollte der Zielknoten vom Netzwerk isoliert sein, erreicht ihn die Nachricht auch auf diesem Wege nicht. Der Postman vermerkt, dass die Nachricht an den Knoten nicht ausgeliefert wurde. Selbiges gilt auch, wenn der Postman keine Bestätigung über den Erhalt bekommt. Dies kann eventuell trotz erfolgreicher Übermittlung passieren, falls der Postman nicht mehr in Reichweite des Netzwerkes des Zielknotens sein sollte.

Fremder Knoten meldet sich beim Postman an Dies kann ein Knoten sein, der tatsächlich nicht zum eigenen Projekt gehört oder aber auch ein fehlerhafter Knoten. Da der Postman dies nicht eigenständig entscheiden kann, sendet er dem Knoten das Signal zur Übermittlung der Nachrichten, speichert sie zusammen mit der MAC-Adresse des Knotens separat in Quarantäne und markiert diese Pakete, damit an der Basis entschieden werden kann, ob sie sinnvoll sind oder nicht. So ist dennoch sichergestellt, dass keine Nachricht verloren geht.

Ein fremder Postman In diesem Szenario befindet sich ein zweiter bzw. fremder Postman in dem Messgebiet. Hier muss im Vorfeld entschieden werden, wie vertraulich die Messdaten der Knoten und die an sie gerichteten Nachrichten sind. Sind die Daten unter allen Umständen vor dem Zugriff Dritter zu schützen, ist allerdings das Komplette hier skizzierte Szenario schwer haltbar, denn dann muss generell eine gesicherte Authentifizierung erfolgen, welche das hier beschriebene Nachrichtenformat nicht vorsieht. Eine Möglichkeit wäre, die Nachrichten zu verschlüsseln, wobei die privaten Schlüssel nur der Basis beziehungsweise jeweils dem einzelnen Knoten bekannt sind. So könnte ein Dritter zwar mitlesen, die Nachricht aber nicht verstehen. Sofern dieses Vorgehen verwendet wird, ist das hier geschilderte Szenario kein Problem, denn dann sind die Nachrichten vor dem Zugriff unberechtigter Dritter geschützt und das normale Vorgehen kann stattfinden.

Ein zweiter Postman im selben Gebiet ist problematischer, denn wenn sowohl dieser als auch der eigentliche Postman unterschiedlichen Knoten das Signal zum Übermitteln der Daten geben, ist eine zuverlässige Übertragung nicht mehr sichergestellt. Einer der beiden muss also wie ein Knoten behandelt werden und abwarten, bis alle Knoten auf

den Befehl des einen Postman gesendet und empfangen haben. Erst danach kann dieser Postman senden und empfangen. Dies ist allerdings eine ziemlich unbefriedigende Lösung, es sollte generell sichergestellt werden, dass dieses Szenario nicht eintritt und sich immer nur ein Postman zurzeit in einem Gebiet aufhält.

Zu klären bleibt allerdings, sowohl bei Einsatz einer Verschlüsselung als auch bei einer Reihung mehrerer Postman, wie mit den Nachrichten der Knoten generell verfahren werden soll. Werden diese, um Speicher zu sparen, nach erfolgter Übermittlung an einen Postman nämlich vom Knoten gelöscht, wären sie in diesem Fall zwar vor dem fremden Zugriff geschützt, aber dennoch verloren, wenn sie nicht an den richtigen Postman übermittelt wurden.

6.7 Ausblick

Im Rahmen dieser Projektarbeit wurden im Bereich des Postman viele relevante Punkte nur angekratzt. Im vorherigen Abschnitt wurden mögliche Komplikationen geschildert und textuell gelöst. Diese müssen sowohl modelliert als auch eingehender betrachtet werden. Gerade das Thema der Nachrichtensicherheit und Vermeidung von Redundanzen, wobei gleichzeitig keine Nachricht verloren gehen darf, ist für die Zukunft ein sehr interessanter Bereich, der im Rahmen weiterer Forschung untersucht werden sollte.

Im gewählten Anwendungsfall haben wir uns mit einem Szenario unter Wasser beschäftigt. Die modellierten Prozesse sind allerdings auch in vielen anderen Bereichen, wie der Car2Car-Kommunikation oder Anwendungen im Weltall einsetzbar. In weiteren Arbeiten sollte gerade auch dieser Aspekt geprüft werden.

7 Task Handover

Michael Recker

7.1 Einleitung

In diesem Teil des Projekts geht es um das Ermitteln, Modellieren und Untersuchen der Fehlertoleranz von Auftragsübergaben zwischen zwei autonomen Unterwasserfahrzeugen. Solche werden zum Beispiel notwendig, wenn mehrere Fahrzeuge einen großen Einsatz gemeinsam bearbeiten.

7.1.1 Kommunikation und Detektion in der Tiefsee

Die absolute Finsternis in der Tiefsee macht eine Kommunikation und Navigation mit in anderen Bereichen verbreiteten elektromagnetischen Radar-, Röntgen- oder Gammastrahlen unmöglich. Alleine die Akustik ermöglicht es, Distanzen von über 100 Meter zu Zwecken der Detektion, Navigation und Kommunikation zu überbrücken. Hierbei sind allerdings sowohl die Zuverlässigkeit als auch die Größe der zu übermittelnden Nachrichten stark eingeschränkt.
Ein konkretes oft verwendetes akustisches Instrument ist das *Synthetic Aperture Sonar (SAS)*, welches mit Hilfe von Pings aus verschiedenen Positionen ein sehr exaktes Bild bzw. eine hochauflösende Kartierung errechnen kann (siehe zum Beispiel Hansen [2011]).

7.1.2 Suche auf dem Meeresboden

Diese Technik kommt zum Beispiel auch zum Einsatz, wenn etwas in den Tiefen der Meere verloren geht - zum Beispiel ein Flugzeug. Zur systematischen Suche in der Tiefsee verschollener Objekte eignen sich insbesondere *Autonomous Underwater Vehicle (AUV)*, da sie nicht den, die hochsensiblen Berechnungen störenden, Oberflächenwellen ausgesetzt sind. Gerade bei großen Einsätzen, ist es gewünscht, mehrere AUVs gemeinsam einzusetzen. Ziel ist es, mit den AUVs ein *mobiles Ad-hoc-Netzwerk (MANET)* zu bilden. Hierbei fungieren die einzelnen AUVs dann sowohl als Sender und Empfänger, aber auch

auch als Relay, bei der Nachrichtenübermittlung.

7.1.3 Koordinierung in autonomen Teams

Um ein optimales *SONAR-Bild* zu erhalten, ist es notwendig, dass die AUVs einen bestimmten - antennenbedingten - Abstand zum Meeresboden einhalten. Das heißt, dass die *Mäander-Pattern* nicht starr sind, sondern vom autonomen Fahrzeug selbst geplant, durchgeführt und kontrolliert werden müssen. Folglich ist die Fahrzeit eines Einsatzes nicht von vornherein bekannt und das AUV muss bei Bedarf entsprechend selbstständig ein weiteres AUV rufen (hierbei muss unter anderem der Standort übermittelt werden) und insbesondere die aktuellen Daten des Einsatzes (zum Beispiel noch ausstehende Suchpattern, Diagonalpunkte und Winkel) an das Ersatzfahrzeug weitergeben. Hierbei spricht man dann von einem *Task Handover*.
Soll beispielsweise ein bestimmtes größeres Gebiet mit mehreren autonomen Unterwasserfahrzeugen nach einem abgestürzten Flugzeug abgesucht werden, müssen die eingesetzten AUVs nach einiger Zeit mit neuer Energie versorgt, inspiziert oder repariert werden. Hierzu ist es notwendig, dass die Fahrzeuge ihre aktuelle Position verlassen. Damit die Suche jedoch fortgeführt werden kann, muss das AUV entsprechenden Ersatz rufen und alle Daten des Einsatzes, wie zum Beispiel den noch abzusuchenden bzw. zu kartierenden Bereich, an den Nachfolger übergeben (*selbstständige Auftragsgenerierung*).

7.1.4 Ausblick

Im Folgenden dieser Arbeit sollen Prozessabläufe für Task Handovers modelliert und insbesondere unter dem Aspekt der Fehlertoleranz überprüft werden. Zur Modellierung verwendet wird *Subject-oriented Business Process Modelling (S-BPM)*, welches es ermöglicht die Beziehungen von Subjekten zueinander und die gegenseitig zu erbringenden Funktionen darzustellen. Eine weitere wichtige Möglichkeit von S-BPM ist die Option zur automatischen Umwandlung des theoretischen Modells in eine ausführbare Form. Dies erlaubt insbesondere die Simulation der erstellten Modellierung und somit auch ein genaueres Untersuchen der Fehlertoleranz. Das heißt konkret zu ermitteln, in wie weit durch fehlerhafte Nachrichten die grundsätzliche Auftragsübernahme, aber auch das richtige Fortführen der übertragenen Aktivitäten, gestört wird. Da S-BPM in Kapitel 2.3 detailliert und anschaulich an einem Beispiel vorgestellt wurde, wird in diesem Kapitel auf eine ausführliche Einführung in das Modellierungswerkzeug verzichtet.

Weiterhin sollen auch byzantinische Fehler untersucht werden. Diese liegen im konkreten Anwendungsbeispiel dann vor, wenn ein AUV Nachrichten an ein oder mehrere weitere autonome Unterwasserfahrzeuge sendet, welche formal korrekt sind - also nicht gegen das Protokoll verstoßen - jedoch falsche Informationen enthalten. Ziel hierbei ist es, derartige Nachrichten zu erkennen (im einfachsten Fall zum Beispiel im Rahmen einer Plausibilitätsprüfung) und entsprechend zu klassifizieren bzw. zu handhaben.
Grundsätzlich wird es bei der Betrachtung von Fehlertoleranzen unumgänglich sein, verschiedene Fehlerarten bzw. Fehlertypen zu unterscheiden.

7.2 Erste Prozessformulierung

In diesem Abschnitt sollen erste Ideen zu *Task Handover Prozessen* formuliert werden. Dies geschieht in natürlicher Sprache und ohne konkrete formale Vorgaben einer Syntax. In späteren Abschnitten dieser Arbeit werden die hier formulierten Prozesse wieder aufgegriffen und stetig erweitert und insbesondere auch formalisiert.

7.2.1 Definitionen

Ein *Scan* am Meeresboden sei definiert durch Anfangs- und Endpunkt sowie den nordbezogenen Winkel der zugehörigen Verbindungsgeraden.

- Anfangspunkt: z.B. $54.350144°, 10.160379°$ (Linke untere Parallelogrammecke)
- Endpunkt: z.B. $54.374238°, 10.176430°$ (rechte obere Parallelogrammecke)
- Winkel (nordbezogen): z.B. $21.21°$

Es gibt weiterhin zwei verschiedene Fahrzeugtypen. Ein *Scan-AUV*, welches mit einem SAS-Instrument ausgestattet den Meeresboden scannt, und ein *Inspektor-AUV*, das bei Bedarf gerufen werden kann und mit einer optischen Kamera ausgestattet ist. Das Scan-AUV generiert selbstständig Mäander-Pattern abhängig von der für den Einsatz des SAS notwendigen Tauchtiefe.
Weiterhin gibt es eine Basisstation (zum Beispiel auf einem Schiff), welche die AUVs bei der Koordination unterstützt sowie ein Netzwerk weiterer Subjekte, welche die Kommunikation unter Wasser ermöglichen.

7.2.2 Prozesselemente

Im Folgenden sollen wesentliche Prozesselemente eines Suchprozesses auf dem Meeresboden formuliert und spezifiziert werden.

Einsatzstart Die Suche wird begonnen, indem ein Scan-AUV ausgestattet mit einem SAS-Instrument und mindestens den drei notwendigen Daten (Anfangs- & Endpunkt, Winkel) in das abzusuchende Wasser gesetzt wird.

Suchvorgang Dem Einsatzstart schließt sich der Suchvorgang an. Während diesem sucht das Scan-AUV unter selbständiger Berechnung der notwendigen Mäander-Pattern den Meeresboden ab. Hierbei werden insbesondere vom Insepktor-AUV näher zu untersuchende Punkte gespeichert.

Einsatzende Beim Suchvorgang kann ein Ereignis eintreten, welches einen Abbruch der Suche durch das betroffene Scan-AUV notwendig macht.

- **Endpunkt erreicht** Hat das Scan-AUV das Ziel - also den in den Auftragsdaten spezifizierten Endpunk - erreicht, ist der Gesamteinsatz abgeschlossen und das AUV kann ebenfalls direkt zur Basisstation zurückkehren.
- **Energiemangel** Das Scan-AUV verfügt nicht mehr über ausreichend Energie und muss die Suche abbrechen. In diesem Fall muss ein weiteres Scan-AUV gerufen und der aktuelle Einsatz übergeben werden. Das abzulösende Scan-AUV muss nun zurück zur Basisstation fahren und mit neuer Energie versorgt werden. Wichtig hierbei ist, dass das Scan-AUV während der fortlaufenden Planung des gesamten Einsatzes immer eine ausreichende Menge an Energie für den Rückweg einplant.
- **Inspektion notwendig** Stellt das Scan-AUV selbst einen Fehler fest oder treten vergleichbare Ereignisse ein, die eine Inspektion an der Basisstation notwendig machen, muss auch hier ein weiteres Scan-AUV gerufen werden, welches den Einsatz fortführt. Analog zum Fall Energiemangel sucht das Scan-AUV nun die Basisstation auf und wird gewartet.

Die beiden hier vorstellten möglichen Szenarien für ein Einsatzende bei noch nicht vollständig abgeschlossenem Auftrag, also *Energiemangel* und *Inspektion notwendig*, stellen nur eine Auswahl vorstellbarer Ereignisse dar und werden aus Gründen der Allgemeinheit im folgenden der Arbeit aggregiert betrachtet.

Task Handover (Auftragsübergabe) Wenn ein Scan-AUV festgestellt hat, dass es den Einsatz nicht fortführen kann, der Gesamteinsatz jedoch noch nicht abgeschlossen ist, findet eine Auftragsübergabe statt. Hierzu muss das Scan-AUV zuerst die neue Scan-Area berechnen, da es einen Teil des ursprünglichen Auftrags bereits bearbeitet hat. Details zu dieser Berechnung werden im folgenden Kapitel beschrieben. Nachdem die neue Scan-Area berechnet wurde, kann das AUV nun einen Nachfolger rufen und nach Erhalt einer End-zu-End-Bestätigung (auch *End-to-End-Acknowledgment*) zur Basis fahren. Der Auftrag wird nun vom ersetzenden Scan-AUV fortgeführt.

Inspektor-AUV rufen Immer wenn das Scan-AUV einen interessanten Punkt, welcher näher untersucht werden soll, findet, soll ein Inspektor-AUV gerufen werden. Diesem wird dann der zu untersuchende Punkt übermittelt.

Die einzelnen Prozesselemente des Inspektor-AUVs wurden hier aus Gründen der Übersichtlichkeit erst einmal ausgeblendet. Grundsätzlich verhalten sich die notwendigen Funktionen und Abläufe jedoch ähnlich wie bei dem Scan-AUV.

7.2.3 Prozessabläufe

Mit Hilfe der entwickelten Prozesselemente sollen nun beispielhaft einige mögliche Abläufe erstellt werden.

Einfache Suche $Einsatzstart \rightarrow Suchvorgang \rightarrow Einsatzende$
Im einfachsten Fall kann ein gesamter Einsatz von einem einzigen AUV abschließend ausgeführt werden.

Suche mit Energiemangel $Einsatzstart \rightarrow Suche \rightarrow Task\ Handover \rightarrow Einsatzende$
Kann ein Scan-AUV einen begonnen Suchvorgang aufgrund von Energiemangel nicht beenden, muss der Einsatz an ein anderes AUV übergaben werden, bevor das Scan-AUV die Basisstation aufsucht.

Suche mit Fund $Einsatzstart \rightarrow Suche \rightarrow Inspektor - AUV\ rufen \rightarrow Suche \rightarrow Einsatzende$
Findet das Such-AUV einen zu untersuchenden Punkt, ruft es ein Inspektor-AUV und setzt unvermindert den Suchvorgang fort.

Bei diesen drei Abläufen handelt es sich um die Grundbausteine möglicher Szenarien in sehr einfacher Darstellung. Denkbar sind auch komplexere Kombinationen, wie im folgenden angedeutet.

Suche mit zwei AUVs $Einsatzstart_1 \rightarrow Suchvorgang_1 \rightarrow Task\ Handover_{1\rightarrow 2} \rightarrow Einsatzende_1/Einsatzstart_2 \rightarrow Suchvorgang_2 \rightarrow Einsatzende_2$
In diesem Fall wird der Suchauftrag von AUV_1 an AUV_2 weitergegeben und anschließend durch dieses beendet.

Durch Erweiterung erhält man ein Szenario, indem sich zwei Scan-AUVs einen Einsatz teilen. Das heißt während AUV_1 den Meeresboden absucht, wird AUV_2 an der Basisstation gewartet oder mit Energie versorgt. Liegt bei AUV_1 nun ein Grund für ein individuelles Einsatzende vor, wird AUV_2 verständigt und im Rahmen eines Task Handovers in den

aktuellen Stand des Auftrags eingewiesen. Dies geschieht abwechselnd solange, bis der Gesamteinsatz von einem der beiden AUVs abgeschlossen werden kann.

Durch kontinuierliche Verallgemeinerung lässt sich das angeführte Szenario weiter verallgemeinern.

7.3 Komplikationen bei Task Handovers

Aufgrund der relativ unzuverlässigen Kommunikationsmöglichkeiten unter Wasser ist ein besonderes Augenmerk auf die möglichen Komplikationen zu legen, welche durch falsche (siehe auch Kapitel 7.6), nur teilweise oder gar nicht empfangene Nachrichten entstehen können.

7.3.1 Checksummen

Alle Nachrichtenpakete enthalten Checksummen zur Überprüfung der Korrektheit. Folglich gibt es drei mögliche Zustände.

- Nachricht syntaktisch **gültig**
- Nachricht **fehlerhaft** (Details siehe 2.4)
- **keine** Nachricht

Eine Checksumme sagt nichts über die semantische Korrektheit einer Nachricht aus. Diese muss gesondert geprüft werden.

7.3.2 Erste Modellierung in BPMN

Im folgenden soll nun ein erster Task Handover Prozess in *BPMN (Business Process Model and Notation)* modelliert werden, um mögliche Fehlerfälle systematisch aufdecken zu können. Dies erleichtert die sich anschließende Analyse und Entwicklung von Möglichkeiten zur Behandlung der potentiellen Fehler. Bei der Modellierung handelt es sich jedoch nur um eine erste Idee, die die Herangehensweise an die Entwicklung von Prozessen verdeutlichen soll, jedoch noch nicht vollständig kompatibel zu den in folgenden Kapiteln vorgestellten Modellen und insbesondere Nachrichtenprotokollen ist.
Hierzu werden zuerst zwei einzelne Prozesse betrachtet. Abbildung 7.1 modelliert das Verhalten des abzulösenden AUVs (AUV_1) bei einem Task Handover. Abbildung 7.2 stellt die Aktivitäten des Ersatz-AUVs (AUV_2) dar. Die konkreten Aktivitäten des MANETS, welche insbesondere bei der Nachrichtenübermittlung eine signifikante Rolle spielen, sollen an dieser Stelle vernachlässigt werden.

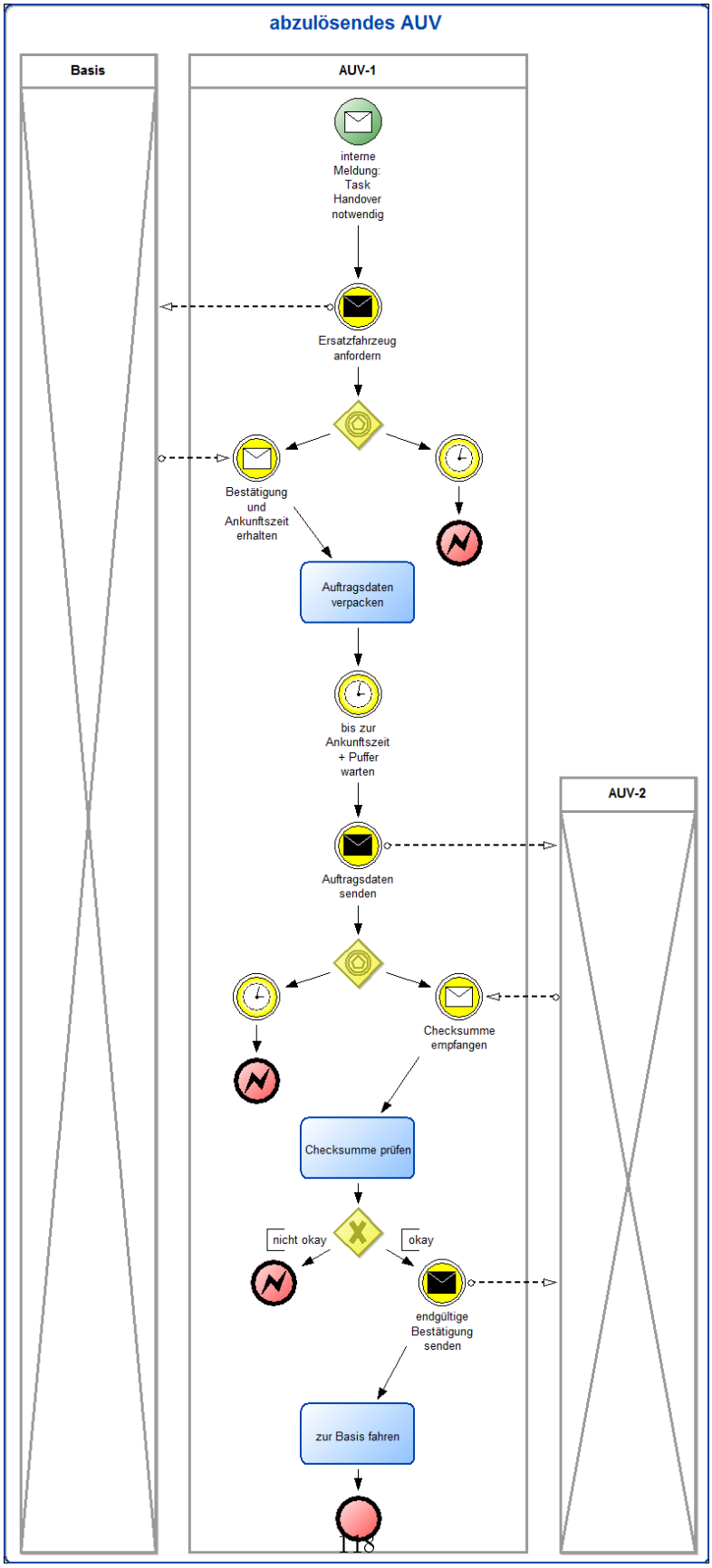


Abbildung 7.1: Prozessablauf des abzulösenden AUVs

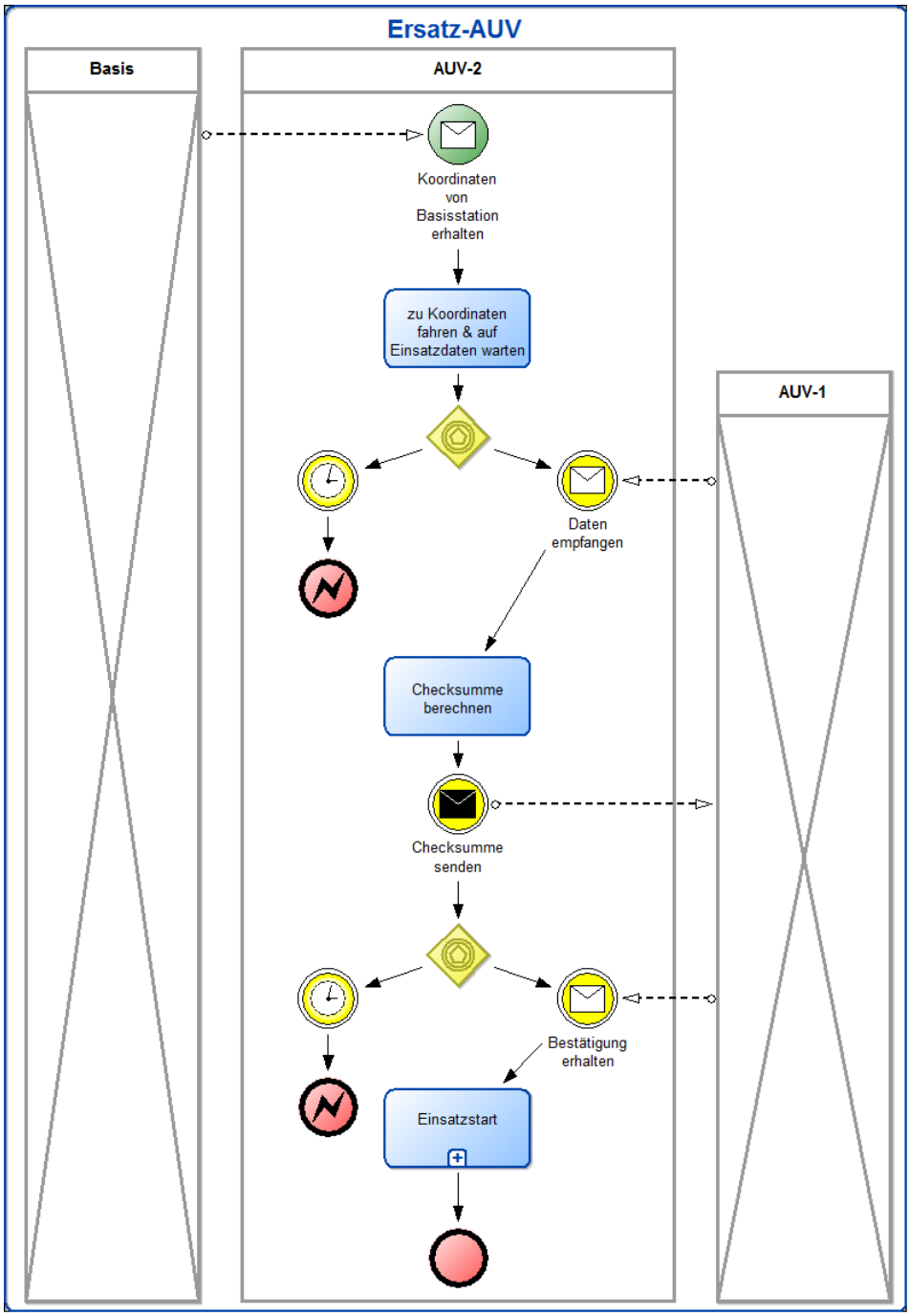


Abbildung 7.2: Prozessablauf des Ersatz-AUVs

7.3.3 Analyse der BPMN-Prozesse

Hier sollen die, in den beiden BPMN-Prozessen durch *Error-End-Events* gekennzeichneten, potentiellen Fehlerfälle analysiert werden. Die meisten dieser Fehler kommen aufgrund von nicht korrekt übertragenen Nachrichten zustande, was in der Tiefsee, jedoch insbesondere auch im Flachwasser, keinen seltenen Sonderfall darstellt.

Abbildung 7.1 (abzulösendes AUV)

- **keine Bestätigung** über angefordertes Ersatz-AUV erhalten: Nachdem das AUV ein Ersatzfahrzeug bei der Basisstation angefordert hat, kann es passieren, dass es keine Bestätigung (und die Ankunftszeit des Ersatz-AUVs) erhält.
- **keine Checksumme** erhalten: Nachdem das AUV die Auftragsdaten an das Ersatz-AUV gesendet hat, kann ein ähnlicher Fall eintreten und das AUV wartet vergebens auf eine Bestätigung in Form einer Checksumme.
- erhaltene **Checksumme nicht korrekt**: Konnte die Checksumme übertragen werden, stimmt jedoch nicht mit der vom zu ersetzenden AUV selbst errechneten überein, liegt ein weiterer Fehlerfall vor. Dieser kann zum Beispiel durch nur partiell korrekt übertragene Daten erklärt werden.

Abbildung 7.2 (Ersatz-AUV)

- **keine Auftragsdaten** empfangen: Nachdem das autonome Fahrzeug das zu ersetzende AUV aufgesucht hat, wartet es auf die Einsatzdaten. Wenn es diese nicht erhält, ist ein Fehlerfall eingetreten.
- **keine Bestätigung** der Checksumme erhalten: Nach der Berechnung und dem Versenden der Checksumme wartet das AUV auf eine endgültige Bestätigung. Im Fehlerfall erhält es diese nicht.

7.3.4 Möglichkeiten zur Fehlerbehandlung

Um die aufgezeigten möglichen Fehler zu umgehen wird im weiteren Verlauf der Arbeit auf folgende Strategie zurückgegriffen:

- Jede Nachricht wird mit einem *impliziten Acknowledgment* versendet, das sicherstellt, dass die Nachricht im Rahmen eines Relays weitergeleitet wurde.

- Einige Nachrichten können mit einem *expliziten Acknowledgment* versendet werden, welches dem Sender das Empfangen der Nachricht beim endgültigen Empfänger (und nicht nur bei einem Relay) bestätigt.
- Erhält ein Sender nicht die gewünschte Bestätigung (*Acknowledgment*), wählt er verschiedene Strategien abhängig von der Relevanz seiner Nachricht. Beispielhaft zu nennen ist hier das mehrfache Senden einer Nachricht (Details siehe Kapitel 3), das verzögerte erneute Versenden oder das erneute Versenden nach einem Positionswechsel.

7.4 Nachrichten im MANET

7.4.1 Generic UnderWater Application Language

Für die Kommunikation der AUVs untereinander soll die *Generic UnderWater Application Language* (kurz *GUWAL*) (Nissen and Goetz [2014]) verwendet werden.
Unterschieden wird hier zwischen zwei GUWAL-Nachrichtentypen, der *Move-to-Nachricht* und der *Inspector-Call-Nachricht*. Beide werden im folgenden kurz vorgestellt, um die betrachteten Prozesse auch auf Protokollebene in einen Gesamtkontext einordnen zu können. Auf tiefer gehende Details wird jedoch bewusst verzichtet, da diese Arbeit einen anderen Schwerpunkt setzt.

AUV bewegen

Tabelle 3.1 (Kapitel 3.4) spezifiziert Nachrichten, um ein autonomes Fahrzeug zu Position *(x,y)* und Tiefe t zu bewegen (*Move-to-Nachricht*).

Inspektor-AUV rufen

Tabelle 7.1 spezifiziert Nachrichten, um ein Inspektor-AUV zu Position *(x,y)* und Tiefe t zu bewegen und ein gefundenes Objekt genauer zu untersuchen (*Inspector-Call-Nachricht*).

Position	Bits	Verwendung
0-1	2	Kontrollnachricht
2	1	End-zu-End Bestätigungs-Flag (Acknowledgement flag)
3	1	Prioritätsmarkierung
4-9	6	betriebliche Startadresse
10-15	6	betriebliche Zieladresse
16-19	4	Typ des Datums
20	1	Flag variable Länge
21-22	2	**Kontakttyp**
23-41	19	lokale Zeit
42-44	3	**Objekttyp**
45-48	4	**Größe** (Höhe)
49-52	4	**Größe** (Breite)
53-56	4	**Größe** (Länge)
57	1	Flag: versenkt?
58	1	Flag: treibend?
59-82	24	**Breite der GPS-Zielposition (y)**
83-106	24	**Länge der GPS-Zielposition (x)**
107-111	5	**Zieltiefe (t)**
112-127	16	Checksumme

Tabelle 7.1: Inspector-Call-Nachricht (aus Nissen and Goetz [2014])

7.4.2 Move-to generieren: Berechnung einer neuen Scan-Area

Wenn ein Scan-AUV eine Auftrag nicht beenden kann und an ein weiteres Scan-AUV übergibt, muss eine neue Scan-Area ideal etwas überlappend berechnet werden, da ein Teil des ursprünglichen Auftrages bereits abgearbeitet wurde. Wie in Kapitel 7.2.1 bereits definiert, besteht eine Scan-Area im wesentlichen aus Anfangspunkt (A), Endpunkt (E) und nordbezogenem Winkel (α). Abbildung 7.3 zeigt eine entsprechende Scan-Area.

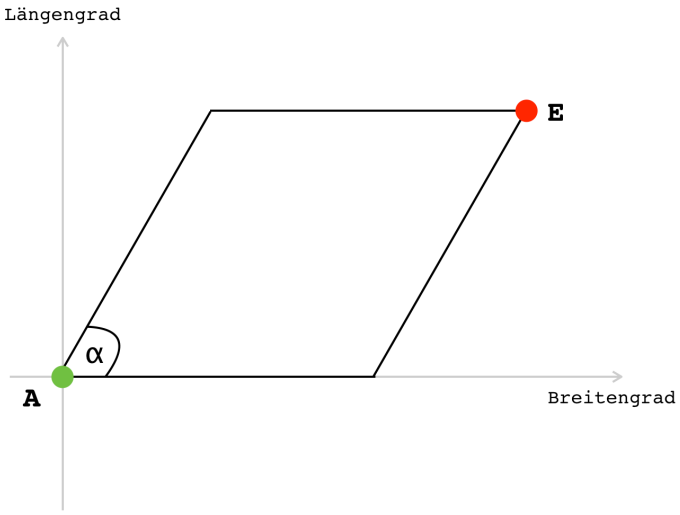


Abbildung 7.3: Scan-Area eines Scan-AUVs

Ein Scan-AUV beginnt die Abarbeitung der Scan-Area an Punkt A und bewegt sich wie in Abbildung 7.4 anhand der gelben Pfeile dargestellt, bis zum Endpunkt E. Kann das AUV den Auftrag vor Erreichen des Endpunktes E nicht fortsetzen (in Abbildung 7.4 ist das z.B. der Punkt X), muss es eine neue Scan-Area berechnen und einem weiteren Scan-AUV eine entsprechende Move-to-Nachricht senden.

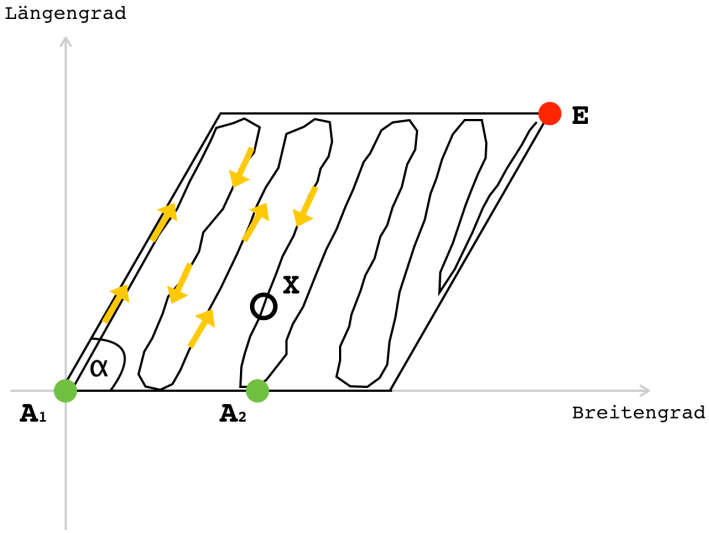


Abbildung 7.4: Abarbeitung der Scan-Area in Mäander-Form

Wie sich Abbildung 7.4 entnehmen lässt, setzt sich die neue Scan-Area aus den Punkten A_2 (Anfangspunkt), E (Endpunkt) und α (Winkel) zusammen. Das heißt im Vergleich zum ursprünglichen Scan-Auftrag muss nur der Längengrad des Anfangspunktes angepasst werden.
Der folgende Algorithmus beschreibt formal, wie die neue Scan-Area durch ein abzulösendes Scan-AUV berechnet wird.

```
calculateNewScanArea(Coordinate A,  Coordinate E, Angel
a, Coordinate
X){

        Coordinate A2 =  new Coordinate(A.getLatitude, X.getLongitude)
        ScanArea newArea = new ScanArea(A2, E, a); return newArea; }
```

Abbildung 7.5: Algorithmus zur Berechnung der neuen Scan-Area

7.4.3 Nachrichtenübertragung - Flooding

Im allgemeinen Fall besteht kein direkter Kontakt der AUVs untereinander und zwischen den AUVs und der Basis. Deshalb müssen die gesendeten Nachrichten über andere Objekte des MANET übertragen werden. Die hierfür benötigten Prozesse und Rahmenbedingungen des Netzwerks werden im Kapitel 3 (Flooding) näher beschrieben.
Im folgenden werden die Prozesse des Netzwerkprotokolls nicht dargestellt, da sie bereits im entsprechenden Kapitel ausführlich beschrieben wurden. Aus Gründen der Übersichtlichkeit werden Nachrichtenbeziehungen abstrahiert und als direkter Kontakt zwischen Sender und Empfänger dargestellt.

7.5 Modellierung in S-BPM

Grundsätzlich soll, wie bereits angedeutet, zwischen zwei verschiedenen Task Handover Events unterschieden werden.

7.5.1 Mögliche Task Handover Szenarien

Scan-AUV → Inspektor-AUV Findet ein Scan-AUV ein zu untersuchendes Objekt, ruft es ein Inspektor-AUV und setzt anschließend direkt seine Scan-Tätigkeit fort (*Fire and Forget*). Für diesen Task Handover Typ wird eine *Inspektor-Call-Nachricht* generiert.

Scan-AUV → Scan-AUV Muss ein Scan-AUV seinen Einsatz aufgrund von mangelnder Energie oder einer anstehenden Inspektion abbrechen, muss ein weiteres Scan-AUV gerufen werden. Dazu berechnet das zu ersetzende AUV mit Hilfe der Einsatzdaten und der bereits zurückgelegten Strecke eine neue Scan-Strecke (gleicher Endpunkt, jedoch neuer Anfangspunkt) und generiert daraus eine *Move-to-Nachricht* für seinen Nachfolger. Nach Erhalt der Bestätigung (auch *Acknowledgement*) fährt das nun ersetzte autonome Fahrzeug zurück zur Basis.

7.5.2 S-BPM Prozessmodelle

Im folgenden Abschnitt sollen nun die bereits oben beschriebenen Task Handover Prozesse unter Beachtung der vorherigen Kapitel mittels S-BPM modelliert werden. Dieses Modell dient in den weiteren Kapiteln dann als Basis für die Untersuchung der Fehlertoleranz.

Akteure in der Kommunikationsschicht

Die Kommunikationsschicht in Abbildung 7.6 zeigt die Nachrichtenbeziehungen der verschiedenen Akteure, auch als *Subjekte* bezeichnet, untereinander auf und bildet die Basis für die Modellierung des Subjekt-internen Verhaltens.
Für die Spezifikation des Task Handovers werden drei Subjekttypen verwendet. Das *hilfsbedürftige Scan-AUV*, welches einen Auftrag ausführt, diesen jedoch nicht zu Ende führen kann und Hilfe rufen muss, sowie das *neue Scan-AUV*, welches den Auftrag vom hilfsbedürftigen AUV übernehmen soll und abstrakt für alle in der Umgebung des hilfsbedürftigen AUVs lokalisierten AUVs steht. Daneben gibt es noch einen weiteren AUV-Typ - das *Inspektor-AUV*, welches Aufträge von Scan-AUVs zugeteilt bekommt und die entsprechenden Punkte näher untersucht. Dieses steht analog zum neues Scan-AUV für

die Menge aller Unterwasserfahrzeuge mit Inspektor-Einheit, welche sich in Bereitschaft befinden.

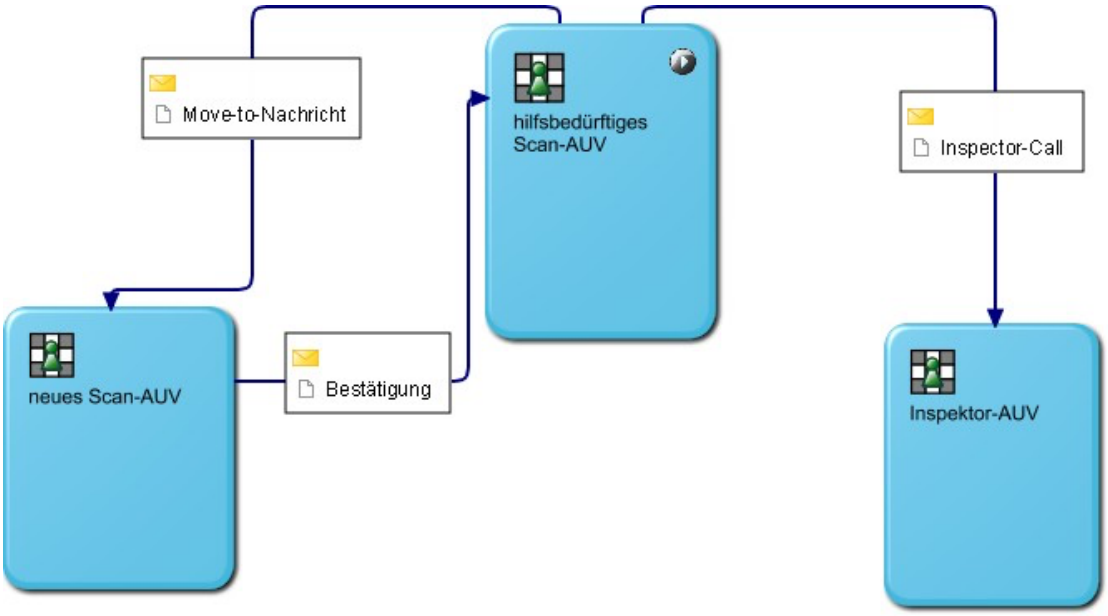


Abbildung 7.6: Kommunikationsbeziehungen der am Prozess beteiligten Subjekte

Die Kommunikationsschicht beschreibt die für die beiden möglichen Task Handover Szenarien notwendigen Nachrichtenbeziehungen. Einerseits kann ein Scan-AUV seine Arbeit an ein anderes Scan-AUV übergeben. Andererseits kann ein Scan-AUV ein Inspektor-AUV rufen, um einen Fund genauer zu untersuchen und somit einen neuen Auftrag für das Inspektor-AUV eröffnen. Die konkret verwendeten Nachrichtentypen wurde bereits ausführlich beschrieben.
Die Move-to-Nachricht wird genutzt, um ein neues Scan-AUV zu rufen, welches den aktuellen eigenen Auftrag übernimmt. Hierbei wird auch der End-zu-End Acknowledgement Flag (*Bestätigung*) verwendet, um sicherzustellen, dass der Auftrag übergeben wurde und weitergeführt wird.
Die Inspektor-Call-Nachricht wird verwendet, um eine Fundstelle an ein Inspektor-AUV zu übergeben und von diesem näher untersuchen zu lassen. Hierbei wird nur die vom Protokoll bzw. des Netzwerks getragene Funktion des impliziten Acknowledgements genutzt und auf eine aufwendigere explizite Bestätigung verzichtet.

Internes Subjektverhalten der drei Akteure

Hilfsbedürftiges Scan-AUV (Abbildung 7.7) Ein Scan-AUV führt zu Beginn des Prozesses einen Auftrag aus, der durch eine Move-to-Nachricht gestartet wurde. Nun wird zwischen drei möglichen Fällen unterschieden:

- **Auftrag erledigt:** Nachdem ein Scan-AUV einen Auftrag vollständig abgeschlossen hat, fährt es zur Basis. Dieser Fall ist trivial.

- **Problem aufgetreten:** Dieser Fall ist am komplexesten und beschreibt die Situation, in der ein Scan-AUV den aktuellen Auftrag nicht fortführen kann. Möglich Gründe hierfür sind zum Beispiel ein Mangel an Energie oder ein Defekt. Nun muss zuerst die neue Scan-Area berechnet werden und eine entsprechende Move-to-Nachricht generiert werden. Dies wurde bereits in Kapitel 7.4.2 ausführlich beschreiben. Nachdem die Nachricht, inklusive der Anforderung einer expliziten Bestätigung (*End-to-End-Acknowledgement-Flag aktiviert*) generiert und versendet wurde, wartet das abzulösende Scan-AUV nun auf die Bestätigung eines anderen Scan-AUVs. Sobald es eine Bestätigung erhalten hat, fährt es zur Basis und der Task Handover wurde erfolgreich abgeschlossen.
  Sollte das alte Scan-AUV keine Bestätigung erhalten haben, wendet es in dem hier modellierten Prozess zwei Strategien an. Einerseits wird die Nachricht mehrfach (3x) gesendet und jeweils drei Minuten auf eine Bestätigung gewartet. Andererseits wird, sollte das mehrfache Senden nicht zum Erhalten einer Bestätigung führen, mehrfach (3x) die Position gewechselt, die Nachricht erneut gesendet und wieder jeweils auf eine explizite Bestätigung gewartet.
  Sollten auch das nicht dazu führen, dass ein neues Scan-AUV den Auftrag übernimmt und dies explizit bestätigt, generiert das abzulösende Scan-AUV einen Fehlerbericht und fährt zur Basis. An dieser Stelle ist nun menschliches Handeln notwendig und der autonome Ablauf wurde unterbrochen. In Kapitel 7.6 wird auf weitere Strategien eingegangen, die ein menschliches Eingreifen noch weniger wahrscheinlich machen und somit eine noch höhere Ausfallsicherheit der Autonomie des Gesamtsystems gewährleisten.

- **zu untersuchendes Objekt gefunden:** Hat das Scan-AUV ein näher zu untersuchendes Objekt gefunden, sendet es eine Inspektor-Call-Nachricht an ein Inspektor-AUV und fährt direkt nach erfolgtem Versand der Nachricht mit seinem Scan-Auftrag fort. Auf eine explizite Bestätigung wird in diesem Fall annahmegemäß nicht gewartet.

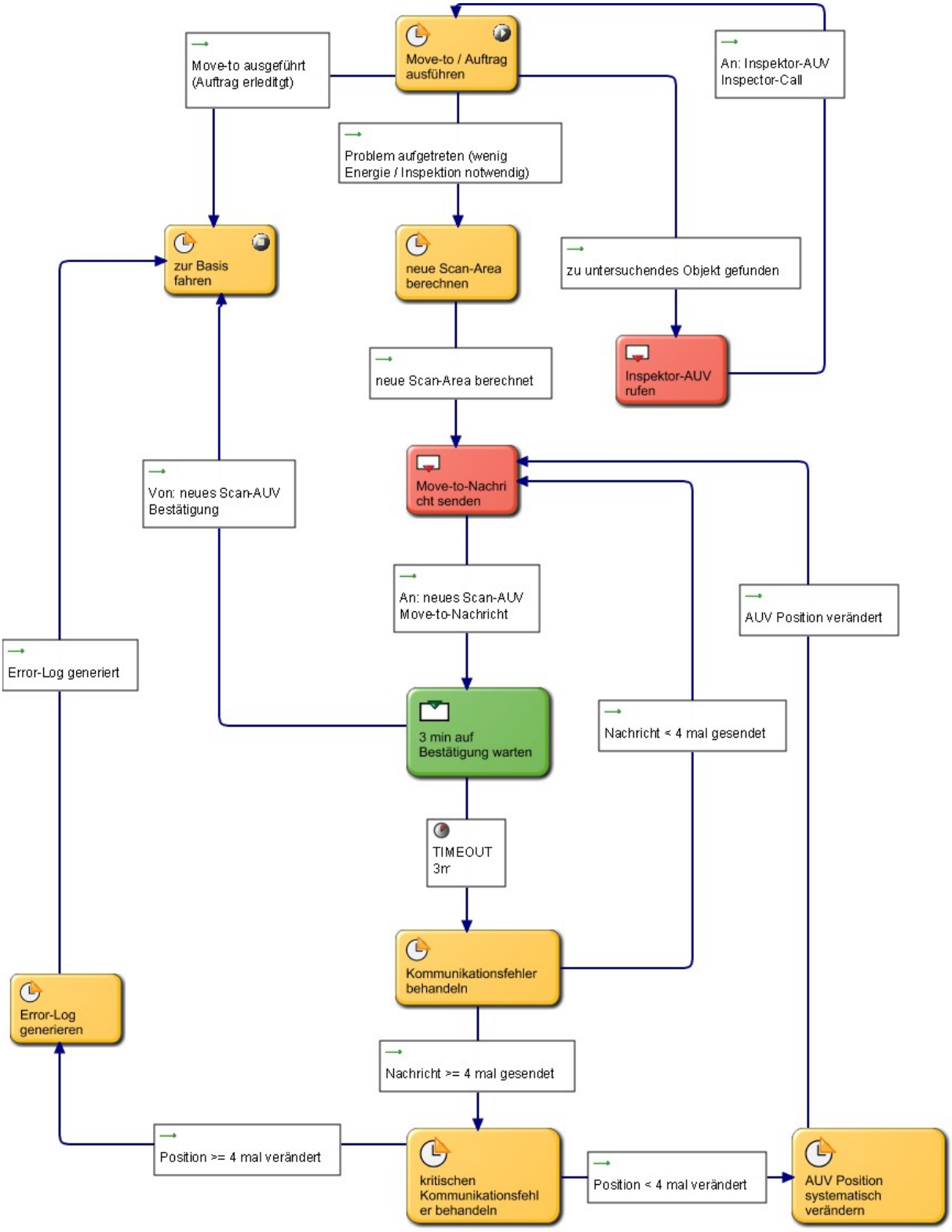


Abbildung 7.7: Internes Verhalten eines hilfsbedürftigen/abzulösenden Scan-AUVs

Neues Scan-AUV (Abbildung 7.8) Die Nachricht des abzulösenden Scan-AUVs richtete sich an alle verfügbaren Scan-AUVs, das heißt solche, die gerade keinen Auftrag ausführen. Daraus folgt, dass der Prozess eines neuen Scan-AUVs mit dem Warten auf einen Auftrag, also einen Nachrichteneingang, beginnt. Als Ausnahme wurde modelliert, dass ein Scan-AUV bei mangelnder Energie die Warteposition verlässt und die Basis aufsucht.
Erhält ein wartendes Scan-AUV nun einen Auftrag, prüft es diesen auf Plausibilität, um unter anderem das Auftreten byzantinischer Fehler zu verhindern. Ist der Auftrag plausibel wird er angenommen, indem eine explizite Bestätigung gesendet und anschließend die entsprechende Scan-Area untersucht wird. Sollte der Auftrag nicht plausibel sein, wird nicht auf die Nachricht reagiert. Das sendende AUV hat annahmegemäß sicherzustellen, dass der Auftrag erfolgreich übergeben wird.

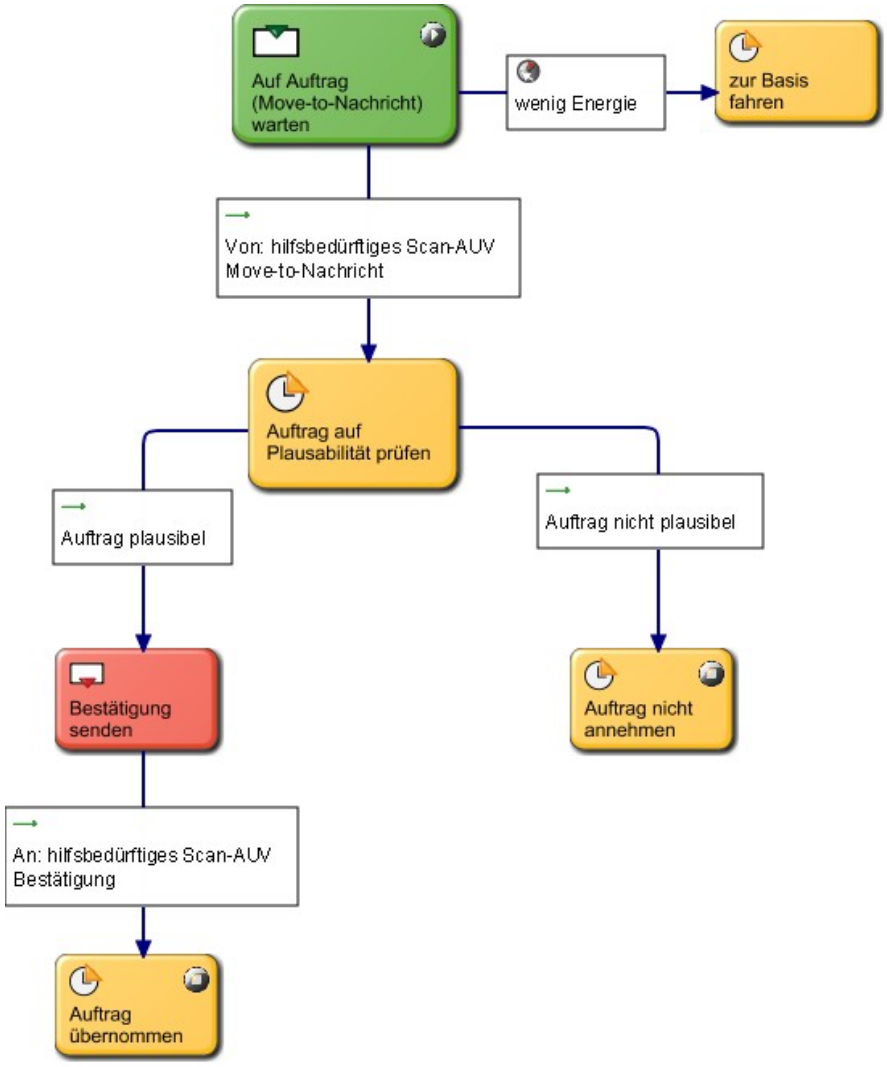


Abbildung 7.8: Internes Verhalten eines neuen/ablösenden Scan-AUVs

Inspektor-AUV (Abbildung 7.9) Ein verfügbares Inspektor-AUV befindet sich zu Prozessbeginn - analog zum neuen Scan-AUV - in einer Wartepostion. Diese wird verlassen, wenn Energie nachgeladen werden muss oder wenn ein Auftrag in Form einer Inspector-Call-Nachricht empfangen wird. Im zweiten Fall sucht das Inspector-AUV das zu untersuchende Objekt auf und inspiziert es mit einer optischen Kamera. Anschließend kehrt es in seine Warteposition zurück und wartete auf weitere Aufträge.
Die Details der Weiterverarbeitung der Ergebnisse des Inspektor-AUVs wurden an dieser Stelle bewusst ausgeblendet, um nicht vom eigentlichen Fokus - der Auftragsübergabe - abzulenken.

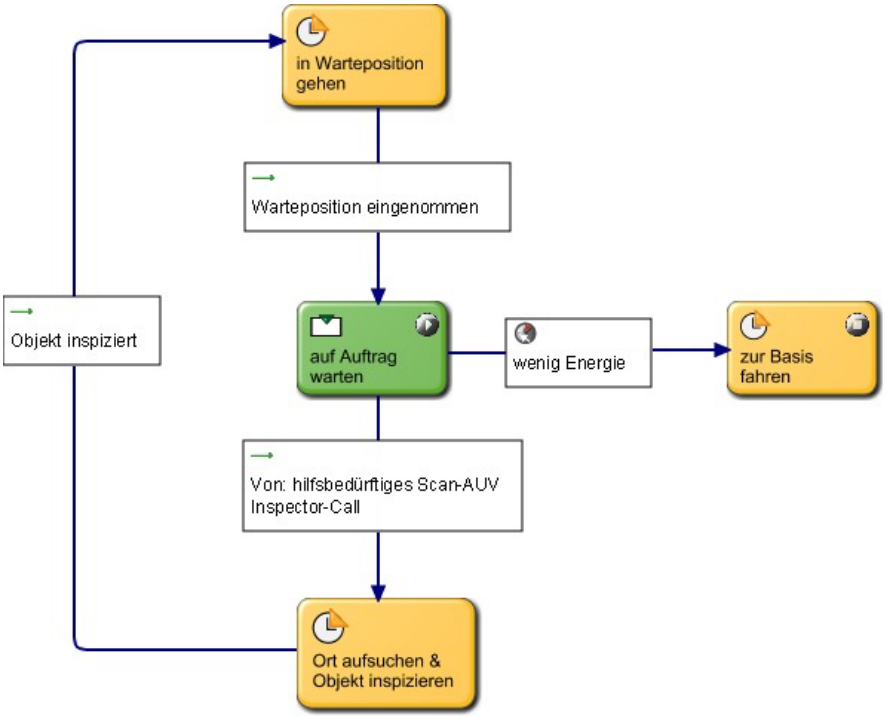


Abbildung 7.9: Internes Verhalten eines Inspektor-AUVs

7.6 Fehlertoleranz

In Kapitel 2.4 wurden unterschiedliche Fehlerarten allgemein und ausführlich beschrieben. Auch die Relevanz der Betrachtung von Fehlertoleranzen in verteilten Systemen wurde dargestellt. Diese spielt auch bei Auftragsübergaben eine übergeordnete Rolle, da es sicherzustellen gilt, dass Aufträge unter den Bedingungen der Tiefsee sicher übergeben werden können. Problematische Einflussfaktoren sind in diesem Gebiet insbesondere die erschwerte Kommunikation mittels Akustik sowie die hohen Kosten, die bei Auftreten von Fehlern und Komplikationen entstehen können.
In diesem Abschnitt sollen nun potentielle Fehlerfälle gefunden werden, die während den vorgestellten Task Handover Szenarien auftreten können. Dies dient einerseits dazu, die Fehlertoleranz der modellierten Prozesse aufzuzeigen, andererseits jedoch auch dazu, um auf die Grenzen der entworfenen Prozessmodelle hinzuweisen und somit eine Basis für weitere Entwicklungen in diesem Bereich zu bilden.

7.6.1 Fehlerbehandlung in modellierten Prozessen

An dieser Stelle soll anhand einiger Beispiele gezeigt werden, wie die entwickelten Prozesse auf Fehler reagieren.

Nachrichtenverlust durch technische Störung

Potentieller Fehler Die Akustik, die in der Tiefsee zur Kommunikation verwendet wird, ist sehr empfindlich. Es ist somit keine Seltenheit, dass Nachrichten verloren gehen. Denkbar wäre folglich ein Szenario, indem ein Scan-AUV ein Inspektor-AUV ruft, die gesendete Nachricht jedoch verloren geht und somit kein Inspektor-AUV den Auftrag ausführt.

Modellierte Lösung Auf Netzwerkebene werden alle Nachrichten dreifach gesendet, sodass die dargestellten Prozesse einen teilweisen Verlust von Nachrichten tolerieren.

AUV erhält keine End-zu-End-Bestätigung

Potentieller Fehler Wenn ein Scan-AUV einen Auftrag an ein anderes Scan-AUV übergeben möchte, lässt es sich die Übernahme vom neuen Scan-AUV mit einer expliziten End-zu-End-Bestätigung versichern. Möglich wäre ein Fall, indem ein Scan-AUV trotz dreifachem Versand der Nachricht auf Netzwerkebene keine solche Bestätigung erhält. Ursächlich hierfür könnten verschiedene Fehlertypen sein. Denkbar wäre zum Beispiel

eine *Empfangsauslassung (Receive omission)* beim abzulösenden AUV. Unwahrscheinlicher, jedoch nicht gänzlich auszuschließen, wäre ein Fehler bei allen potentiellen neuen Scan-AUVs - zum Beispiel ein *Dienstausfall (Omission Failure)*.

Modellierte Lösung Im formulierten Prozess in Abbildung 7.7 wird das Scan-AUV die Nachricht nun erneut versenden und anschließend mehrfach aus verschiedenen Positionen senden. Sollte selbst dann noch keine Bestätigung eingegangen sein, fährt das Scan-AUV zur Basis und ein Mensch muss in das autonome System eingreifen und die Auftragsübergabe manuell durchführen. Dieses Beispiel zeigt sehr anschaulich, dass es nahezu unmöglich ist, eine hundertprozentige Fehlertoleranz in einem autonomen verteilten System mit realistischem Aufwand sicherzustellen.

7.6.2 Mögliche Lösungen für nicht behandelte Fehler

Im folgenden sollen nun beispielhaft mögliche Fehlerfälle skizziert werden, um die Grenzen der Fehlertoleranz der modellierten Prozesse aufzuzeigen.

Nicht erreichbare Koordinate

Potentieller Fehler Vorstellbar wäre ein Szenario in dem ein Scan-AUV einen Move-to-Auftrag erhält, welcher eine Koordinate enthält, die sich auf der anderen Seite des Globus befindet - vom Scan-AUV also nicht erreichbar ist. Hierbei handelt es sich um einen *byzantinischen Fehler*. Die Nachricht an sich ist korrekt, das heißt insbesondere die Koordinate ist syntaktisch richtig spezifiziert. Semantisch ist diese Nachricht jedoch offensichtlich falsch. In den in dieser Arbeit spezifizierten Prozessen würde sich das Scan-AUV dennoch auf den Weg machen und lange Zeit vor erreichen des Ziels im Ozean aufgrund von Energiemangel in der Tiefsee verloren gehen.

Mögliche Lösung Grundsätzlich sollte vor jeder Annahme eines Auftrages eine Plausibilitätsprüfung vom AUV vorgenommen werden. So müsste hier beispielsweise nur geprüft werden, ob sich die erhaltene Koordinate innerhalb des Bereichs befindet, in dem das MANET operiert. Diese Prüfung wurde in Abbildung 7.8 bereits angedeutet.
Weiterhin ist es auch für andere Szenarien sehr zu empfehlen, dass AUV prüfen zu lassen, ob es einen Auftragsort überhaupt erreichen kann - auch wenn dieser innerhalb des bekannten MANETs liegt. Hierzu muss das AUV berechnen, ob es die Zielkoordinate und den Rückweg zur Basis zuzüglich eines Energiepuffers von zum Beispiel 20% in einer gegebenen Geschwindigkeit erreichen kann. Nur wenn das der Fall ist, darf der Auftrag angenommen werden.

Fehlerhaft koordinierte Antwortintervalle

Potentieller Fehler Ein Scan-AUV könnte eine Move-to-Auftrag mit einem End-zu-End-Flag versenden. Das heißt nach Versenden der Nachricht wartet das AUV auf eine explizite Bestätigung des endgültigen Empfängers. Braucht der Empfänger jedoch länger um zu antworten, als das Scan-AUV auf eine Antwort wartet, kommt es zu einem *zeitbedingten Ausfall (Timing Failure)*. Dieser würde in den modellierten Prozesse dazu führen, dass der Auftrag zwar ausgeführt, der Sender darüber jedoch nicht informiert wird. Es könnte passieren, dass das sendende AUV den Auftrag ein zweites Mal nun an ein drites beteiligtes Fahrzeug vergibt und der Auftrag doppelt ausgeführt wird.

Mögliche Lösung Der Fehlerfall lässt sich umgehen, indem die Zeitintervalle, in denen auf eine Antwort gewartet wird, auf die Größe des MANETs abgestimmt werden. In den formulierten Prozessen wurde ein Zeitintervall von 3 Minuten gewählt und angenommen, dass dies ausreichend ist. In der Praxis sollte die Wahl der Antwortintervalle jedoch stets wohlüberlegt sein.

7.7 Fazit und Ausblick

Das Kapitel Task Handover dieser Arbeit sollte einen Eindruck von den bei einer Auftragsübergabe in autonomen Teams ablaufenden Prozessen und deren Fehlertoleranz vermitteln.
Nach der Einführung wichtiger Begriffe und Definitionen wurden grundlegende Prozesselemente und -abläufe formuliert, die in späteren Abschnitten immer wieder aufgegriffen wurden. Anschließend wurden einfache Task Handover Prozesse zuerst in BPMN modelliert, um mögliche Fehlerfälle anhand einer verbreiteten Modellierungssyntax aufzuzeigen. Weiterhin wurde kurz auf die technischen Grundlagen auf Netzwerk- und Protokollebene eingegangen, um das Kapitel in den Gesamtkontext einzubetten. Kern der Arbeit stellte die darauf folgende Modellierung in S-BPM dar, welche erstmals Prozesse für die Auftragsübergabe autonomer Fahrzeuge unter Wasser spezifiziert. Die Modelle wurden im folgenden auf Fehlertoleranz untersucht und auch auf noch nicht behandelte potentielle Fehlerfälle wurde eingegangen.

Da es sich bei dem behandelten Thema um einen neuen, wenig untersuchten Bereich handelt, wurden in dieser Arbeit grundlegende Funktionen und Abläufe modelliert, welche die Basis nahezu jeder Auftragsübergabe unter Wasser darstellen. Die formulierten Prozesse können als Basis für spezifischere Modellierungen dienen, aber auch an sich weiter verfeinert werden - insbesondere in Hinblick auf die Fehlertoleranz, wie in Kapitel 7.6 bereits angedeutet.

8 Danksagung

Wir danken Dr. Ivor Nissen für seine hervorragende Unterstützung bei der Umsetzung unserer Ideen und für die zur Verfügung gestellten Materialien. Des Weiteren bedanken wir uns bei Metasonic für die Möglichkeit, ihre Metasonic Suite kostenfrei für die Modellierung unserer Prozesse zu nutzen. Darüber hinaus bedanken wir uns bei Prof. Dr. Bernhard Thalheim, dass er uns dieses Projekt bei Herrn Nissen ermöglicht hat.

9 Kooperation zur dezentralen Datenfusion

Ivor Nissen

Eine weitere Kooperationsform in Unterwassernetzwerken besteht bei der Datenfusion von Sensorergebnissen. Eine Verlagerung von Verantwortung an die beteiligten Subjekte, hier die Sensorknoten, zieht eine vereinfachte Koordinierung nach sich, wie es bei der Auftragstaktik her bekannt ist. Anhand der Peilungsbestimmung soll eine dezentrale Kooperation vorgestellt und diskutiert werden.

Gegeben sei ein Unterwasserknoten mit akustischer Sensorik, der mittels mehrerer Hydrofone und Beamforming die Elevation und den Azimut (nordbezogen) zu einem Objekt feststellen kann. Das ist beispielsweise in der Biologie in Form von Tracking von maritimen Säugern eine anspruchsvolle Aufgabe, um das Wanderverhalten der Wale feststellen zu können. Ein Bodenknoten (beispielsweise mit mindestens vier thetraförmig angeorgneten Hydrofonen oder einem Vektorhydrofon) kennt seine eigene Position (letzte GPS-Fix vor dem Wassereintritt) und generiert ein Detektionsereignis in Form einer Peilungsmeldung.

A Generieren einer Peilungsmeldung (z. B. als GUWAL-Telegramm). Durch den eingebauten Kompass sei die genutzte Peilung nordbezogen.

Benachbarte Bodenknoten leiten diese Nachricht mittels des Netzwerkprotokolls „ungelesen" weiter (siehe Kapitel 3). Die relativen Peilungsmeldungen unterschiedlicher Bodenknoten werden in einer Zentrale im Netzwerk in zeitlicher Reihenfolge ausgewertet. Unter Wasser ist zwar eine Peilungsbestimmung je nach Anordnung und Anzahl der Hydrofone relativ genau schätzbar, eine Entfernungsbestimmung ist passiv aus zwei Gründen jedoch nur sehr grob schätzbar: 1) Der Ausbreitungsverlust (Transmission Loss, TL) inklusive der Absorption im Wasserkörper ist eine nichtlineare Funktion in Ort, speziell der Tiefe, und Frequenz, im Flachwasser durch die Mehrwegeausbreitung zudem schwer

zu modellieren. 2) kennt man oft nicht die absoluten Sendepegel der aufgefassten Signale (die Geräuschpegel der Wale), um mit dem geschätzten Ausbreitungsverlust auf die Entfernung zu schließen. Für die Entfernungsbestimmung bedarf es der Kooperation und es sei auf Kapitel 5 verwiesen. Durch die geringe Schallgeschwindigkeit im Wasser, im Vergleich zur Lichtgeschwindigkeit in Luft ist noch ein weiteres Problem anzusprechen: Bis die Zentrale alle Peilungsmeldungen eingesammelt und ausgewertet hat, ist Zeit in der Größenordnung von Minuten für eine Reaktion vergangen, kommt noch hinzu, das durch die ungünstigen Ausbreitungsbedingungen nicht jede Nachricht in der Sensordatenfusionszentrale ankommt, werden die Schwächen dieser zentralistischen Form deutlich.

In Nissen et al. [2015] wurde daher eine dezentrale Strategie vorgeschlagen, die man aus der Auftragstaktik her kennt und die auf einer engen Kooperation aufbaut. Ist man beispielsweise an dem Mittelwert einer Messgröße wie dem Energiezustand im Netzwerk interessiert, so ist die zentralistische Herangehensweise die Zusendung aller Einzelwerte durch das Netzwerk, in der Zentrale wird das arithmetische Mittel als Summe geteilt durch die Anzahl der auch mehrfach eingegangenen Einzelwerte bestimmt und wieder an alle als Broadcast zurückgeflutet. In dem dezentralen Ansatz sendet ein Knoten die Größe und den Zählerstand. Jeder Knoten, der die Nachricht erhält und liest, multipliziert mit dem gelesenen Zählerstand, addiert die zu sendende Messgröße und erhöht den Zählerstand um eins. Dann wird der Wert geteilt durch den neuen Zählerstand mit ihm versendet. Damit ist den Kommunikationsknoten das Wissen über den Mittelwert beim ersten Fluten im Netzwerk bekannt.

$$G(n) = \frac{1}{n}\sum_{i=1}^{n} G_i \text{ analog zu } G(1) = G_1/1; G(i) = ((i-1) * G_{i-1} + G_i)/i, i > 1.$$

Für die Peilungsaufgabe ist der Ablauf analog: Wieder generiert ein Bodenknoten eine Detektionsmeldung mit der Peilung (A). Ein Nachbarknoten empfängt die Nachricht und meldet mittels den Regeln in Kapitel 3 weiter. Zusätzlich wertet der Bodenknoten die empfangene Nachricht aus, er „liest die Postkarte mit". Die Netzwerkregel sei: Alle Nachrichtentypen die von unmittelbaren Nachbarn eingehen und die ein Knoten auch selber generieren könnte (Fremdwissen), werden zusätzlich an die eigene Anwendung intern weitergeleitet, auch wenn sie nicht für einen selbst bestimmt sind. In diesem Fall der Peilungsmeldung würde damit die Detektionsanwendung nicht nur die eigenen Detektionsereignisse erarbeiten, sondern über Nachbardetektionsereignisse mitinformiert die relativen Peilungen des Umfeldes kennen. Jeder Bodenknoten hört seine unmittel-

baren Nachbarn (1-hop Nachbarschaft) und kann bei stationären Knoten deren Position erfragen.

B Eine fremde Detektionsmeldung aus der 1-Hop-Nachbarschaft geht ein (wird vom Netzwerkprotokoll intern an die Detektionsanwendung weitergeleitet). Ist die Meldung älter als 5 Minuten, verwerfe diese. Wurde in den letzten 5+1 Minuten ebenfalls eine eigene Detektionsmeldung generiert und ausgesendet? Nein: breche ab, Ja: mache weiter und setze letzte Peilung ownElevation[1]

C Ist die Position von diesem Nachbar-Knoten bekannt? Nein: Frage nach Position (request), warte auf Antwort, fahre bei „Ja" fort oder breche ab. Ja: Generiere durch Kreuzpeilung aus beiden Positionen und beiden nordbezogenen Peilungen eine Kontaktmeldung in Form des Schnittpunktes, wie nachfolgend im Quellkode angegeben.

D Ist der errechnete Schnittpunkt im Detektionsradius enthalten (Distanz zum eigenen Standpunkt innerhalb der Detektionsentfernung), dann melde den Kontakt mit seiner vermuteten Position durch Generierung der Detektionsmeldung.

Eine Javascript-Großkreis-Bestimmung könnte bei Eingabe von Latitude, Longitude und Bearing von Knoten 1 und 2 den Schnittpunkt der Peilungsgerade wie folgt bestimmen:

```
var theta13     = Bear1*Math.PI/180.0;
var theta23     = Bear2*Math.PI/180.0;
var deltaPhi    = (Lat2-Lat1)*Math.PI/180.0;
var deltaLambda = (Lon2-Lon1)*Math.PI/180.0;
var delta12     = 2.0*Math.asin(Math.sqrt(
Math.sin(deltaPhi/2.0)*Math.sin(deltaPhi/2.0)+
Math.cos(Lat1*Math.PI/180.0)*Math.cos(Lat2*Math.PI/180.0)*
Math.sin(deltaLambda/2.0)*Math.sin(deltaLambda/2.0)));

if (delta12 == 0) return null; // initial/final bearings between points

var thetaa      = Math.acos( (Math.sin(Lat2*Math.PI/180.0)                                    -
Math.sin(Lat1*Math.PI/180.0)*Math.cos(delta12)) / (
Math.sin(delta12)*Math.cos(Lat1*Math.PI/180.0)));

if (isNaN(thetaa)) thetaa = 0; // protect against rounding
var thetab      = Math.acos(( Math.sin(Lat1*Math.PI/180.0) -
Math.sin(Lat2*Math.PI/180.0)*Math.cos(delta12)) / (
Math.sin(delta12)*Math.cos(Lat2*Math.PI/180.0) ));

var theta12    =  Math.sin((Lon2-Lon1)*Math.PI/180.0)>0 ? thetaa : 2.0*Math.PI-thetaa;
var theta21    =  Math.sin((Lon2-Lon1)*Math.PI/180.0)>0 ? 2.0*Math.PI-thetab : thetab;

var alpha1     = (theta13 - theta12 + Math.PI) % (2.0*Math.PI) - Math.PI; // angle 2-1-3
var alpha2     = (theta21 - theta23 + Math.PI) % (2.0*Math.PI) - Math.PI; // angle 1-2-3

if  (Math.sin(alpha1)==0 &&  Math.sin(alpha2)==0) return null; // infinite intersections
if  (Math.sin(alpha1)*Math.sin(alpha2) < 0)       return null; // ambiguous intersection
```

[1] man beachte, dass der Wal in der Zwischenzeit natürlich weiter schwimmt, dies ist für diese Prinziperklärung hier nicht substanziell und wird vernachlässigt.

```
var alpha3     = Math.acos(  -Math.cos(alpha1)*Math.cos(alpha2) +
Math.sin(alpha1)*Math.sin(alpha2)*Math.cos(delta12));
var delta13    = Math.atan2( Math.sin(delta12)*Math.sin(alpha1)*Math.sin(alpha2),
Math.cos(alpha2)+Math.cos(alpha1)*Math.cos(alpha3));
var phi3       = Math.asin( Math.sin(Lat1*Math.PI/180.0)*Math.cos(delta13) +
Math.cos(Lat1*Math.PI/180.0)*Math.sin(delta13)*Math.cos(theta13));
var deltaLambda13 = Math.atan2( Math.sin(theta13)*Math.sin(delta13)*Math.cos(Lat1*Math.PI/180.0),
Math.cos(delta13)-Math.sin(Lat1*Math.PI/180.0)*Math.sin(phi3));
var lambda3    = Lon1*Math.PI/180.0 + deltaLambda13;
document.write("Lat:    "); document.write(phi3/Math.PI*180.0);
document.write("<br>");
document.write("Lon:    "); document.write((lambda3/Math.PI*180.0+540)%360-180);
```

Es kann sich eine Quick-Look-Identifizierung an diesen Prozess anschließen. Auch hier können Identifizierungsmerkmale automatisiert zusammengefügt werden. Nachbarknoten erhalten diese Kontaktmeldung mit Position und können Geschwindigkeiten und Tracks approximieren. Zudem kann durch Einbeziehung der Höhe ein Validierung stattfinden. Dieser dezentrale Ansatz führt ebenfalls zu einem lernenden MANET und hat den Vorteil, dass aufgrund der langsamen Schallausbreitung schneller reagiert werden kann, als gegenüber dem üblichen zentralen Ansatz. Für die nächst höhere Ebene der Koordinierung ergeben sich zudem die aus der Auftragstaktik bekannten Vorzüge.

10 Schlusswort

Ivor Nissen

Die hier vorgestellte Auswahl an Basis-Anwendungen zur Kooperation konnte mittels S-BPM und dem Programm-Werkzeug der Metasonic Suite als eigenständige Ministories simuliert werden. Es wurde gezeigt, welches die für die einzelnen Kooperationen essentiell wichtigen Daten sind und wie Kommunikationsabläufe mit minimalen Nachrichtenübertragungen ablaufen könnten.

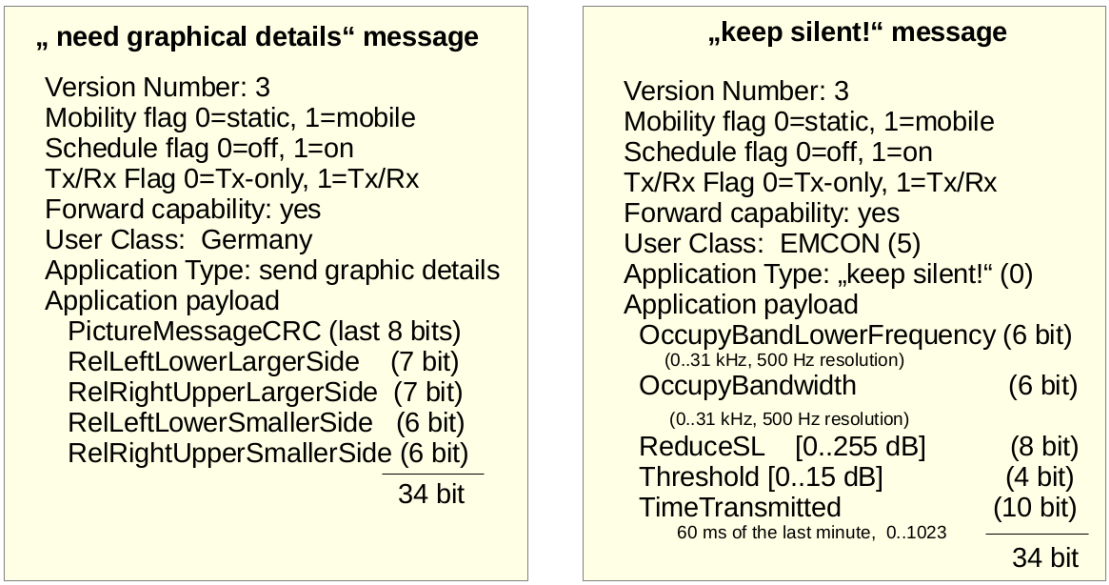


Abbildung: *Bit-Belegungen für die beiden JANUS-Applikations-Payloads zum interaktiven Bildaufbau und Emmison-Control*

Es fehlen weitere Basis-Kooperationen, wie angesprochen zum interaktiven Austausch von Bilddatenausschnitten (siehe Vorwort (1)[1]), „Keep silent!”-Emmissionskontrolle, zum

[1] Gegeben Object.jp2; Low-Layer: j2k_to_image -i Object.jp2 -o ObjectLayer5j2k.tif -l 5; Ausschnitt: convert 'Object.jp2[30x30+435+210]' ObjectPart.png

Emergency-Fall mit interaktivem Zugriff auf Blackbox-Daten (Norddeutsche Studie), Beschreibungen zur Kommunikation mit Offshore-Infrastrukturen als Kollisionswarnung respektive zur Inspektion (Offshore-Gründungen von Wind- und Wellen-Generatoren) sowie nachrichtentragende Beacon-Signale (Fixed Vertical Mooring, Hazard Marker). Da diese Einzelanwendungen nicht singulär, sondern in Gruppen zum „Alltag" eines Kommunikationsteilnehmers gehören, sind diese als „Allgemeinwissen" parallel zu betrachten. So sind Weiterleiten, Distanzmessungen, auf den Postboten reagieren, eine „Begrüßung" in Form eines Erstkontaktes durchführen und vieles mehr, Basisfähigkeiten eines jeden Netwerkteilnehmers.

Wenn davon ausgegangen werden kann, dass die Wahrscheinlichkeit eines Fehlverhaltens eines Gesamtsystems mit der Anzahl der beteiligten auch nebenläufigen Prozesse steigt, sind Konflikte vorprogrammiert. Diese Nebenläufigkeiten und Gesamtprozessketten mit der Nähe zur Implementierung im Sensornetzwerk bedürfen zudem einer Konkretisierung in einer Programmiersprache als Umsetzung der S-BPM-Modellierung. Da die Nachrichtenübertragung von der Bitübertragungsschicht (die Nachrichtenübertragung im Wasser) bis zum Netzwerkprotokoll von vielen Faktoren abhängt, wie beispielsweise dem Seegebiet und dem Wetter, ist nur mit viel Aufwand eine Validierung innerhalb von Seeexperimenten und die Prüfung der Funktionskette inklusive Anwendungen möglich. Es kann dann zwischen Fehlern unterschiedlicher Schichten nicht mehr getrennt werden. Da aus Sicht der Anwendung immer nur korrekte Nachrichten prozessiert werden (übermittelte Prüfwerte stimmen mit Daten überein, CRC korrekt), kann die Anwendungsschicht für sich simuliert werden. Durch die kleinen Übertragungsvolumen von 64 und 128 bit sind entsprechend die Schutzgrößen der Prüfwerte in Form von 8 (HD=8) und 16 bit (HD=10) auch entsprechend klein. Bei einem 64 bit-Container (56 bit Userdaten plus 8 bit CRC) haben wir 2^{56} mögliche Nachrichten, jedoch nur 2^8 unterschiedliche Prüfwerte. Damit haben $2^{56-8} \approx 10^{14}$ Nachrichten den gleichen CRC-Wert. Die Wahrscheinlichkeit von auftretenden Mutationen ist hoch, also syntaktisch korrekten aber sematisch falschen Nachrichten, bei denen Prüfwerte und zugehörige Nachricht zusammenpassen. Dazu ein

Beispiel: Gegeben seien Anwenderdaten von 0 zu 15, die mittels 4 bit repräsentiert werden können. Zum Schutz seien drei weiter Bit spendiert. Ein CRC-3 aus dem GSM-Dokument ETSI TS 100 909 (RC-3, 1011, x^3+x+1, V8.9.0, Sophia Antipolis, 21 October 2016) mit einer Hamming-Distanz von 3 liefert alle gültigen Codewörter:

0, 0000 000 / 1, 0001 011 / 2, 0010 110 / 3, 0011 101 / 4, 0100 111 / 5, 0101 100 / 6, 0110 001 / 7, 0111 010

Alle anderen Kombinationen ($2^7 - 2^4 = 112$) haben CRC-Werte, die nicht mit den vier Anwenderbit übereinstimmen und daher vom Netzwerklayer als inkorrekt abgewiesen werden. Offensichtlich gibt es hier für jeden Prüfwert $2^{4-3} = 2$ gültige Nachrichten, die sich um mindestens 3 bit unterscheiden. Aber auch im Fall nur eines Bitfehlers bei den Anwenderdaten und zwei Bitfehlern bei dem Prüfwert können durch die Bitübertragung in der Netzwerkschicht ebenfalls Nachrichten entstehen, die zwar syntaktisch gültig sind, jedoch nicht der ausgesendeten Nachricht entsprechen - es sind Mutationen entstanden, die bis zur Anwenderschicht vorgedrungen sind. Die Wahrscheinlichkeit bei Gleichverteilung ist nicht vernachlässigbar.

Das kann beabsichtigt sein (Spoofing), aber auch zufällig durch die Bitübertragung geschehen, wenn Datenblock und Prüfwert gleichermaßen passend gestört worden sind. Eine semantische Erkennung ist auf der Anwendungsebene zwingend notwendig, die auch Relaibility-Schätzungen aus der Bitübertragung- und Linkebene mit einschließt. Hier bietet sich die Einbeziehung vom beispielsweise erweiterten Chandy-Lamport Algorithmus an, um globale konsitente Zustände des MANET zu erhalten. **Es wird zwingend eine Simulationsumgebung der Anwenderschicht benötigt.**

Als Ausblick sollen daher in entsprechenden Arbeiten die JANUS- und GUWAL-Anwendungen in Form von Atom-Feeds verarbeiten, wobei ein Planet-Venus-Server-Framework die unterschiedlichen Feeds der Anwendungen einsammelt, verarbeitet und beantwortet. Dabei ist die Identifikationsnummer der Nachricht, gleich ihrem Inhalt. Aus dem vorher angegebenen ist es auch wichtig, auf Anwendungsebene die Vorgeschichte der Nachricht zu kennen, es sind die Prozessierungsschritte der vorangegangenen Ebenen (UWA-PHY/UWA-DL/UWA-NWK/UWA-BUN) gemäß ISO 30140 mitzuliefern.

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
    <author>
        <name>UWA APS - BottomLander #3</name>
     </author>
     <title>JANUS Receiver Version 3</title>
     <id>janus:1234567890123456</id>
     <updated>2016-12-14T10:20:09Z</updated>
     <generator>UnderwaterTelephone</generator>
     <entry>
         <title>to UWA-BUN</title>
         <link href="http://www.is.informatik.uni-kiel.de/research/UWmodel/JANUS"/>
         <id>janus:1234567890123456</id>
         <updated>2016-12-14T10:20:09Z</updated>
         <summary>user class: / app type:</summary>
         <content type="html">
             Version Number<br>
```

```
            Mobility flag 0=static, 1=mobile<br>
            Schedule flag 0=off, 1=on<br>
            Tx/Rx Flag 0=Tx-only, 1=Tx/Rx<br>
            Forward capability yes/no<br>
            User Class <br>
            Application Type<br>
            Application payload<br>
            CRC
        </content>
    </entry>
</feed>
```

Oder es werden GUWAL-Nachrichten ausgetauscht:

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
    <author>
       <name>Gateway buoy #4 (Aquadrone)</name>
    </author>
    <title>Surface Environment Data Parcel</title>
    <id>guwal:4FFF303F4D22B0D2F54EC000B04FE466</id>
    <updated>Mon Aug 13 08:35:33 2012</updated>
    <entry>
        <title>from UWA-BUN</title>
        <link href="http://www.is.informatik.uni-kiel.de/research/UWmodel/GUWAL"/>
        <id>guwal:4FFF303F4D22B0D2F54EC000B04FE466</id>
        <updated>Mon Aug 13 08:30:40 2012</updated>
        <summary type="html">Surface Drift Direction: 23 / Drift Speed: 176 kn / Sea State: 6</summary>
        <content type="html">
           Checksum Correct: 1<br>
           Parcel Type: Data<br>
           Parcel Length: 128<br>
           Acknowledgement: 0<br>
           Priority: LOW<br>
           Source Address: 3.15<br>
           Destination Address: 3.15<br>
           Data Type: Surface Environment<br>
           Data Length: 96<br>
           Variable Length: 0<br>
           Data Length: 91<br>
           Requested Type: 0<br>
           Local Time: Mon Aug 13 08:30:40 2012<br>
           Drift Direction: 23<br>
           Drift Speed: 176 kn<br>
           Sea State: 6<br>
           Rain: 2<br>
           Weather state: 1<br>
           Temperature: 15 C<br>
           IR-Information in air: 679<br>
           Optical Jerloy classes: 6<br>
           MAC: 45135<br>
           Checksum: E466
        </content>
    </entry>
</feed>
```

Letztere gültige Nachricht ist eine Mutation der Ursprungsnachricht guwal:4FF730BB 4922B0D8E55EC000B44BE466 mit einer Hamming-Distanz von 12. Derzeit werden im Kontext der ISO 30140 und dem Internet-of-things entsprechende Schnittstellen auf Basis von Atom-Feeds, MQTT (Message Queue Telemetry Transport), NMEA (National Marine Electronics Association) und CoAP (Constrained Application Protocol) mit Zigbee untersucht und vorgeschlagen. Moglich wären auch Beacons, jedoch sind Atom-Feeds

ausdrucksstärker. So kann auch auf GeoRSS und weitere geospatiale Elemente über <georss:point>38.971684 -92.275641</georss:point> aufgebaut werden. Beispiele sind Discord bots, die Atom-Feeds lesen und beantworten können. Durch eine Simulation können beispielsweise Timing-Fehler entdeckt und abgefangen werden.

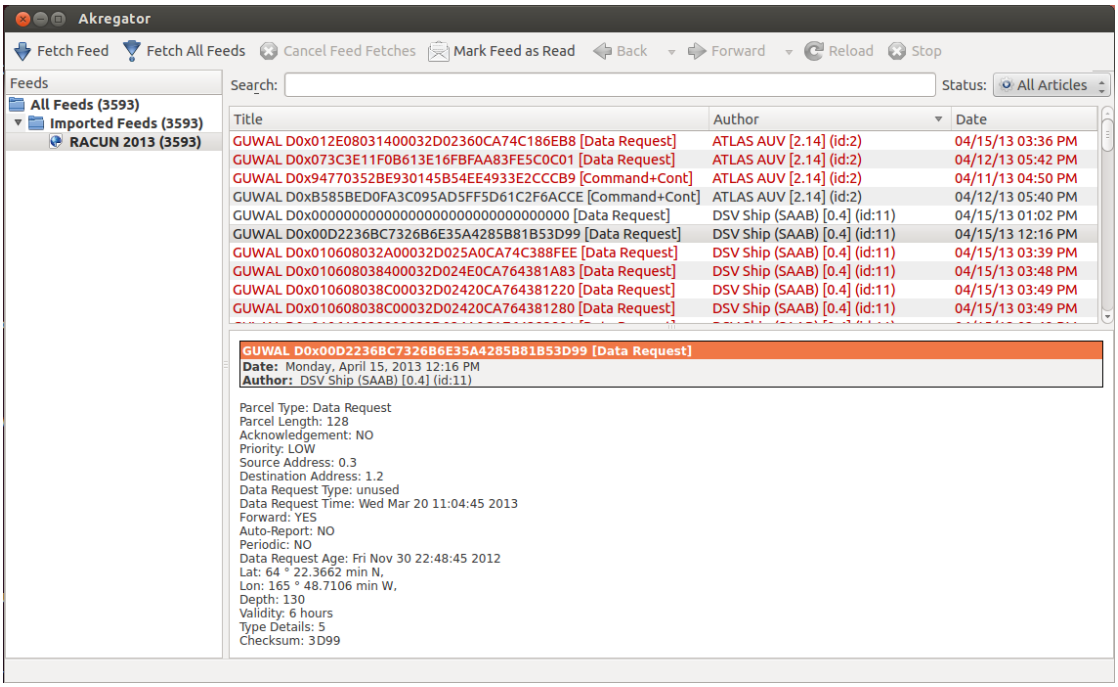


Abbildung: *Atom-Feed-Kollektion aus dem RACUN-Seeversuch 2013 vor Den Helder durch Nutzung der Anwendersprache GUWAL in der Software Akregator*

Es bedarf daher weiterer Untersuchungen und der Erarbeitung eines konsistenten Simulationsframeworks für Unterwasseranwendungen. Dieser Beitrag der Studenten stellt eine erste Annäherung dar.

Literaturverzeichnis

Deadlyhappen. Iso-osi-7-schichten-modell. https://de.wikipedia.org/wiki/OSI-Modell#/media/File:ISO-OSI-7-Schichten-Modell(in_Deutsch).svg, 2014.

R. Guerraoui and A. Schiper. Software-based replication for fault tolerance. *Computer*, 30(4):68–74, Apr. 1997. ISSN 0018-9162. doi: 10.1109/2.585156. URL http://dx.doi.org/10.1109/2.585156.

V. Hadzilacos and S. Toueg. Distributed systems (2nd ed.). chapter Fault-tolerant Broadcasts and Related Problems, pages 97–145. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1993. ISBN 0-201-62427-3. URL http://dl.acm.org/citation.cfm?id=302430.302435.

R. E. Hansen. Introduction to synthetic aperture sonar, sonar systems. InTech, 2011. URL https://www.intechopen.com/books/sonar-systems/introduction-to-synthetic-aperture-sonar.

B. W. Johnson. Fault-tolerant computer system design. chapter An Introduction to the Design and Analysis of Fault-tolerant Systems, pages 1–87. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996. ISBN 0-13-057887-8. URL http://dl.acm.org/citation.cfm?id=230303.230304.

multilateration.com. Multilateration. http://www.multilateration.com/surveillance/multilateration.html. Accessed: 2017-05-02.

NATO. Mxp-1(d)(navy)(air) multi-national submarine and anti-submarine exercise manual. North Atlantic Treaty Organization, 2002.

I. Nissen and M. Goetz. Guwmanet - multicast routing in underwater acoustic networks in military communications and information systems conference. Communications and Information Systems Conference, Gdansk, Poland, 2012.

I. Nissen and M. Goetz. Generic underwater application language (guwal). https://www.researchgate.net/publication/306097796_Generic_Underwater_Application_Language_GUWAL, Mai 2014.

I. Nissen, M. Goetz, and S. Schreiber. Secure underwater coordination of manned and unmanned platforms. *Naval Forces*, 36(4):57–, Sept. 2015.

J. Potter, J. Alves, D. Green, G. Zappa, I. Nissen, and K. McCoy. The janus underwater communications standard. Proc. Conf. Underwater Communications and Networking (UComms), 2014.

A. S. Tanenbaum and M. van Steen. *Verteilte Systeme - Prinzipien und Paradigmen*. Pearson Studium - IT, 2007. ISBN 978-3-8273-7293-2.