Entwicklung und Simulation eines selbstorganisierenden Funknetzwerkes

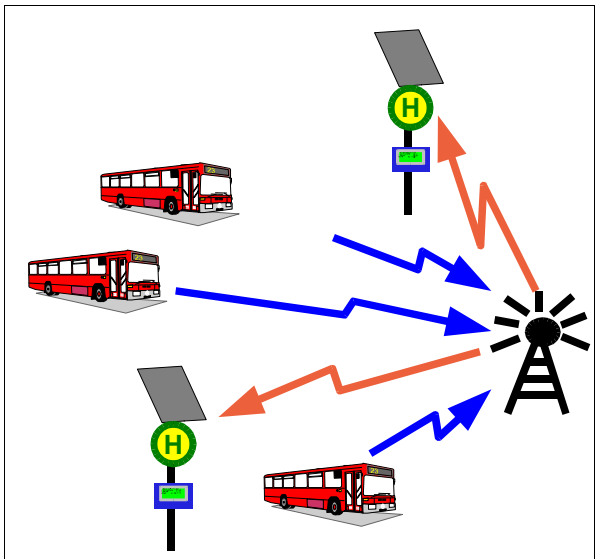
eine Jugend forscht-Arbeit von
Marco Möller und Sebastian U. Brandes

Inhaltsverzeichnis

1 Die Idee

1.1 Die Grundidee

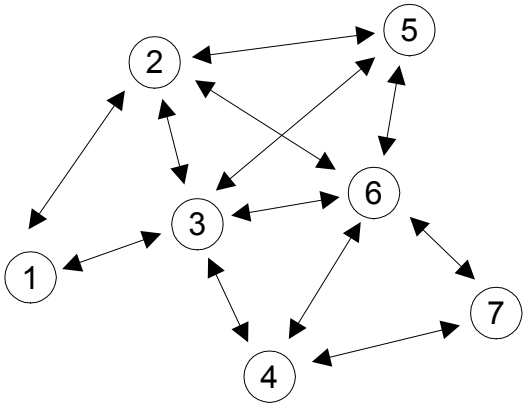
In dem ersten Jugend forscht-Projekt von Björn Eikermann und Marco Möller, dem experimentellen dynamischen Fahrgastinformationssystem, traten einige große Probleme bei der Bereitstellung einer ausreichenden Funkreichweite auf. Das System bestand aus vielen sendenden Stationen (Busse), einer Zentrale und vielen empfangenden Stationen (Haltestellen), die relativ gleichmäßig über ein großes Gebiet verteilt waren. Jede der empfangenden bzw. sendenden Stationen musste einen direkten Funkkontakt zur Zentrale haben. Das Problem bestand darin, dass unsere Funkverbindung nur eine Reichweite von ca. 8 km hatte. Dies ist typisch für ein sternförmiges Funknetzwerk.



Übersicht über die Kommunikation des experimentellen dynamischen Fahrgastinformationssystems

Uns kam damals schon die Idee, die einzelnen Stationen als Repeater einzusetzen, um die Reichweite zu erhöhen. Dazu fehlten uns aber Prozessoren, die die dafür benötigte Leistung zur Verfügung stellen konnten.

Daraus entstand der Gedanke, ein echtes Netzwerk zu gestalten, was in solchen und ähnlichen Fällen seinen Einsatz finden könnte. Dieses Netzwerk sollte im ersten Ausbau für die Telemetriedatenübertragung geeignet sein und das Auffinden der Kommunikationswege sollte vollautomatisch erfolgen, sodass der Benutzer nur noch das Ziel eingeben muss.



Beispiel eines Netzwerkes, in dem alle Kommunikationsmöglichkeiten einzeichnet sind

1.2 Ziele

Im letzten Jahr haben wir versucht Methoden zu erarbeiten, mit denen dieses Netzwerk realisiert werden kann. Im vorherigen Projekt haben wir an der Simulationsumgebung und der Protokollimplementierung gearbeitet, sind damit jedoch nicht fertig geworden, da wir uns mit

dem nötigen Zeitaufwand verschätzt hatten.
Für dieses Jahr hatten wir uns vorgenommen, die Software und das Protokoll komplett zu überarbeiten. Sie sind zu zwei relativ eigenständigen Produkten weiterentwickelt worden: zum Einen die Simulationsplattform und zum Anderen ein komplett ausgearbeitetes, dokumentiertes und implementiertes Protokoll. Dies wurde anschließend noch intensiv erprobt. Hierbei sind Designfehler im Protokoll entdeckt worden, die nun bis zum Landeswettbewerb beseitigt werden. Aus diesen Grund bitten wir um Verständnis, wenn sich im Gegensatz zum hier beschriebenen Protokoll noch ein wenig ändern wird.

2 Arbeitsweise

Im letzten Jahr haben wir uns an gängigen Standards orientiert und uns am System von anderen Protokollen (OSI, TCP/IP, WLAN, etc.) orientiert.

Als es darum ging, das Projekt dieses Jahr weiterzuführen, haben wir erst einmal eine Bestandsaufnahme gemacht, um zu sehen, wie weit wir sind und welche Mängel das Projekt hatte. Daraus entstand der Plan, wie man am besten diese Mängel beseitigen und sowohl das Protokoll als auch die Simulationsplattform verbessern kann.
Zuerst haben wir eine Aufgabenstellung formuliert, aus der wir dann einen Zeitplan erarbeitet haben, in dem vermerkt ist, bis wann was erarbeitet sein sollte. Wir hatten uns vorgenommen, die Arbeitszeit immer mitzuprotokollieren, um jederzeit den aktuellen Stand mit dem Zeitplan abzugleichen. So konnten wir regelmäßig den Zeitplan unserem aktuellen Stand anpassen und sehen, ob wir nicht etwas schneller arbeiten sollten.

Wir haben eine Arbeitsweise entwickelt, in der wir erst etwas auf einem Whiteboard skizzieren, da man so einfach und schnell Dinge korrigieren kann. Anschließend wird das Ganze auf dem PC übernommen, um eine Übersicht zu gewinnen und detailliert Fehler zu beseitigen.

Ein großes Anliegen war es uns im Team zu arbeiten. Dies und die Einhaltung des Zeitplanes war allerdings „dank“ unserer Terminkalender nicht immer einfach!

3 Ergebnis

3.1 Simulationsplattform

Die Simulationsplattform, im folgenden Text „SimPlatt“ genannt, ermöglicht es, die Umgebung, die für das Funknetzwerk benötigt wird, in einigen Bereichen realistisch nachzubilden.

3.1.1 Entstehung

Wir haben uns entschieden, diese Entwicklung einem Hardwareaufbau vorweg zustellen. Fehler lassen sich in einer rechnergestützten Simulation einfacher, schneller und besser finden, da man hier auf die reichhaltige Unterstützung der großen Entwicklungsumgebungen zurückgreifen kann. Leider hat uns (mal wieder) die Zeit für einen Hardwareaufbau gefehlt, weshalb wir diesen von vornherein auch gar nicht geplant haben.

Die SimPlatt bestand bis vier Wochen vor dem Regionalwettbewerb aus einem Server-Client-System. Wir hatten uns für diese Struktur entschieden, da wir dachten, die Bedienung und die Handhabung der Clients zu vereinfachen. Man konnte die Clients auf mehreren Rechnern starten und auf einen Server zugreifen. Infolgedessen konnte man an mehreren Clients parallel arbeiten, ohne das ein „Fensterchaos“ auf einen Monitor entstand. Nachteilig war jedoch, dass eine zentrale Fehlerkorrektur erschwert wurde, da das System nach jeder Korrektur neu gestartet werden musste. Aus diesem Grund haben wir das System kurzfristig umstrukturiert. Nun besteht die SimPlatt nur noch aus einem einzigen Programm. Unsere Bedenken wegen der Handhabbarkeit haben sich bald aufgelöst, da uns neue Bedienkonzepte eingefallen sind.

3.1.2 Die Grundstruktur der SimPlatt

Eine grundlegende Aufgabe der SimPlatt besteht darin, die einzelnen Clients und Stör- und Dämpfungsquellen zu verwalten. Die SimPlatt stellt weiterhin den Funkraum für die Simulation zur Verfügung.

3.1.3 Die Clients

Die Clients sind keine starren Punkte, sondern können sich mit einer gewissen Geschwindigkeit auf einer vorgegebenen Bahn (Polygon) bewegen. Die einzelnen Clients können aktiviert und deaktiviert werden. Zudem kann man zu Testzwecken die Bewegungen global stoppen.

Die Clients sind so strukturiert, dass unser Protokoll als eigene Klasse implementiert ist, sodass man dieses schnell und einfach gegen ein anderes austauschen kann. Die Kommunikation zwischen SimPlatt und Protokoll beschränkt sich auf Sende- und Empfangsbytes, Empfangsfeldstärke, aktueller Sendemodus (RX/TX), Timingsignal und Setzen des Adressbytes.

Für die Protokollklasse(n) wirkt diese Hülle wie ein Funk-IC, den es ansteuert. Das Protokoll bekommt also gar nicht mit, dass es nur in einer Simulation läuft. Zudem lässt das Timingsignal zu, dass die Simulation in verschiedenen Geschwindigkeiten lauffähig ist.

3.1.4 Störungen

Eine andere Aufgabe der SimPlatt ist es, Störungen zu simulieren. Dies wären zum Einen Dämpfungen durch Berge, Wälder, etc., welche als Linienkette mit einer spezifischen Dämpfung angegeben werden. Zum Anderen können auch HF-Störquellen, z. B. Radiosender, Motoren, etc., vorhanden sein, die dann als Punktquelle dargestellt werden. Hier können auch einige Parameter eingestellt werden: Stärke der Störung, Störintervall, Störintervallfluktuation, Stördauer und Stördauerfluktuation. Außerdem wird ein allgegenwärtiges Hintergrundrauschen eingeblendet.

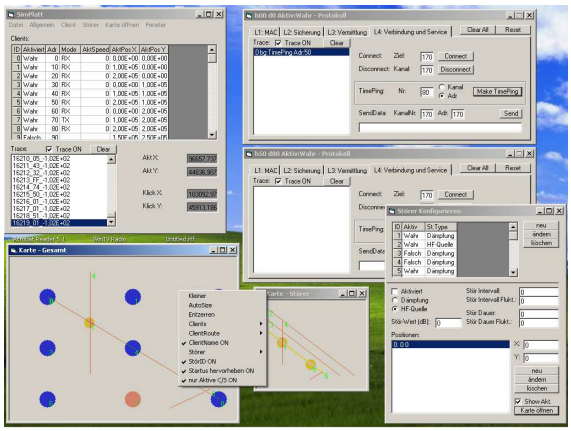
3.1.5 Verarbeitung

Bei der Simulation wiederholen sich im Wesentlichen folgende Schritte:

Falls Clients eine Bewegungsroute besitzen, werden diese zuerst mit der eingegeben Geschwindigkeit bewegt. Als nächstes werden die Störungen je nach Rhythmus aktiviert, bzw. deaktiviert. Nun wird für jeden Client der aktuelle Empfang berechnet. Zur Berechnung der Pegel werden die Entfernungen, die Antennengewinne und die Dämpfungen in der Signalstrecke berücksichtigt. Nun wird der stärkste Signalpegel mit der Summe aus Störungen und den übrigen Signalen in ein Verhältnis gesetzt, in das sogenannte Signal-Rausch-Verhältnis. Danach ändert sich die Wahrscheinlichkeit für ein korrekt übertragendes Datenbyte, bzw. ein Zufallsbyte. Als letztes werden diese Bytes an die Protokolle gesendet und ein weiteres Timingsignal ausgelöst.

3.1.6 Darstellung

Um all diese Funktionen graphisch darzustellen, kann man sich alle Clients, Störquellen und Dämpfungen in einer Karte bzw. mehreren Karten darstellen lassen. Wir haben uns insgesamt bemüht, die Bedienung der Oberfläche so einfach wie möglich zu gestalten. Für jeden Client lässt sich wahlweise ein Formular öffnen, das Zugriff auf diverse Funktionen des Protokolls beinhaltet.



Benutzeroberfläche der SimPlatt in voller Entfaltung

3.2 Protokoll des selbstorganisierenden Funknetzwerkes

3.2.1 Grobe Übersicht

Wir haben versucht, wie letztes Jahr auch, uns am OSI-Schichtmodell zu orientieren. Dabei sind folgende Schichten entstanden:

- Schicht 1: MAC (MediumAccessControl / Mediumzugriffskontrolle)
- Schicht 2: Sicherungsschicht
- Schicht 3: Vermittlungsschicht
- Schicht 4: Verbindungs- und Serviceschicht

Die Kommunikation wird mit Hilfe von Paketen realisiert. Es sind verschiedene Paketgruppen entstanden, die ihre Dienste für die einzelnen Schichten verrichten. Dies ist eine auch bei anderen Protokollen häufig verwendete Vorgehensweise.

3.2.2 Die Pakete

Wir beschreiben nun erst einmal ausführlich den Aufbau der Pakete, um dann in den nächsten Punkten auf ihre Verwendung einzugehen.

3.2.2.1 Aufbau der Pakete

Die Pakete sind in Kopf- und Datenbereich aufgeteilt. Der Kopf hat eine festgelegte Länge sowie Inhalt und definiert durch letzteren die Größe und den Aufbau des Datenbereichs. Kopf und Datenbereich sind jeweils mit Checksummen versehen, um Übertragungsfehler erkennen zu können und um die Wahrscheinlichkeit, dass im Rauschen ein vermeintliches Paket erkannt wird, zu verringern.

Die Wahrscheinlichkeit liegt bei 1/256, dass ein Kopf fälschlicherweise im Rauschen erkannt wird, wenn sein Beginn nur durch das Startbyte definiert ist. In Kombination mit der Checksumme sinkt diese Wahrscheinlichkeit schon auf 1 / 65536. Dadurch, dass im Kopf bei einigen Bytes nur bestimmte Werte zugelassen sind, sinkt diese Wahrscheinlichkeit noch einmal um den Faktor 58. Falls wir nun weißes Rauschen mit 1000 Bytes/s auf einen Kopf untersuchen würden, hätten wir so nur ca. alle Stunde fälschlicherweise einen Kopf erkannt. Die Wahrscheinlichkeit, dass dieser „falsche“ Kopf zu einen fälschlicherweise erkannten Paket wird, wird durch die beiden Checksummenbytes im Datenbereich noch einmal um den Faktor 65536 gesenkt. So dass wir nun alle 8 Jahre, (Fehlerwahrscheinlichkeit 1 zu $2{,}5 * 10^{11}$!!!), bei Dauerbetrieb, ein Problem mit dem vorhin beschriebenen weißen Rauschen hätten. Dazu muss man allerdings wissen, dass dieses weiße Rauschen in der HF (Hochfrequenz)

schon durch manche Funk-ICs unterdrückt wird und nur in Ausnahmesituationen als Datenrauschen auftritt.

3.2.2.2 Der Paketkopf

Bytebezeichnung	Startbyte	Größe des Datenbereichs	Pakettyp	Von wem	An wen	Checksumme
Beispieldaten	FFh	2Bh	5Fh	E5h	4Ah	91h

Der Paketkopf hat eine definierte Länge von 6 Bytes:

- Das erste Byte bildet das Startbyte mit dem festen Wert FFh (255) und erleichtert es, den Beginn des Kopfes einfacher festzustellen.
- Das nächste Byte gibt an, wie viele Bytes der Datenbereich inklusive Checksumme enthält. Somit sind wir in der Lage, den Datenbereich der Auslastung des Paketes anzupassen und somit nicht verwendete Bandbreite freizugeben.
- Das dritte Byte gibt den Pakettyp an. Hiermit wird festgestellt, wie der Datenbereich zu deuten ist und somit wo und wie es verarbeitet werden muss (siehe Tabelle unter 3.2.2.3).
- Im vierten Byte steht die Absender-
- und im fünften Byte die Empfängeradresse.
- Den Schluss bildet dann im sechsten Byte die Checksumme. Sie wird gebildet, indem die fünf vorangegangenen Bytes bitweise XOR-verknüpft werden.

3.2.2.3 Der Datenbereich der Paket

In der nachfolgende Tabelle sind alle Pakete mit der Aufschlüsselung des Pakettypbytes sowie deren Datenbereichen aufgelistet.

Im Pakettypbyte erkennt man anhand der ersten beiden Bits, in welcher Schicht dies Paket seine Verwendung findet (00 = Schicht 2, 01 = Schicht 3, 10 = Schicht 4).

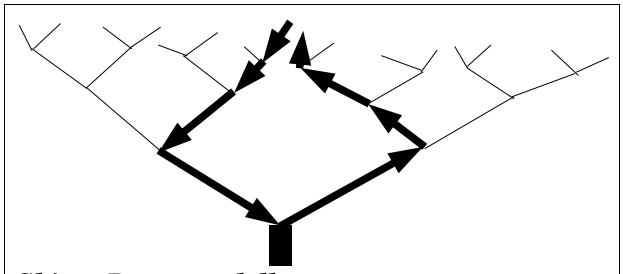
In der Spalte über den Aufbau des Datenbereichs kann man erkennen, wie die Reihenfolge der Bytes im Datenbereich des entsprechenden Paketes ist, indem man die einzelnen Felder mit ihren Byteanzahlangaben hintereinander stellt. Die Checksumme wird ähnlich wie im Kopf gebildet, mit der Ausnahme, dass immer zwei Bytes miteinander verknüpft werden, um eine Zwei-Byte große Checksumme zu erhalten. Bei einer ungeraden Bytezahl wird ein Null-Byte (00h) ergänzt.

Beim Regionalwettbewerb wurden wir auf folgende Missverständlichkeit hingewiesen. Bei den Adressbytes im Datenbereich (Adresse des Gesuchten / Suchenden, Quelle, Endziel) handelt es sich um die Adressen der beiden Enden der Kommunikationskette. Die Adressbytes

des Senders und Empfängers im Kopf hingegen sind die direkten Kommunikationspartner, also zwei benachbarte Punkte in der Weiterleitungskette.

Paketname	**Pakettypbyte im Kopf**	**Aufbau des Datenbereichs (Bytes in dieser Reihenfolge)**
OK-Pakete	00h (0000 0000)	*Kein Datenbereich*
Lokaler Suchping	40h (0100 0000)	• Adresse des Gesuchten (1 Byte) • Checksumme (2 Byte)
Lokaler Antwortping	41h (0100 0001)	• Adresse des Gesuchten (1 Byte) • Anzahl der Weiterleitung bis Erreichen (1 Byte) (Unbekannt: Weiterleitungen = 255) • Checksumme (2 Byte)
Globaler Suchping	42h (0100 0010)	• Adresse des Gesuchten (Ziel, 1 Byte) • Adresse des Suchenden (Quelle, 1 Byte) • Anzahl der Weiterleitung bis Erreichen (1 Byte) • Pingnummer (1 Byte) • Liste der Adressen der weiterleitenden Stationen First (neben PingNr.) = Letzte Station (0 - 40 Byte) • Checksumme (2 Byte)
Globaler Antwortping	43h (0100 0011)	• Adresse des Gesuchten (Quelle, 1 Byte) • Anzahl der Weiterleitung bis Erreichen (1 Byte) • Pingnummer (1 Byte) • Liste der Adressen der weiterleitenden Stationen First (neben PingNr.) = Letzte Station (0 - 40 Byte) • Checksumme (2 Byte)
L4 Pakete	8?h (0100 ????)	• Quelle (1 Byte) • Endziel (1 Byte) • Paketnummer (1 Byte) • Anzahl der Weiterleitungen (1 Byte) • Daten (0-98 Byte) • Checksumme (2 Byte)
Datenpakete	80h (0100 0000)	• Siehe L4 Pakete • PaketstreamPaketNr (1 Byte) • Daten (0-97 Byte)
Laufzeitanfrage	81h (1000 0001)	• Siehe L4 Pakete • im Datenbereich leer
Laufzeitantwort	82h (1000 0010)	• Siehe L4 Pakete • im Datenbereich leer
Connect-Wunsch	83h (1000 0011)	• Siehe L4 Pakete • im Datenbereich: Connect/Disconnect (1 Byte)
Connect-Quittierung	84h (1000 0100)	• Siehe L4 Pakete • im Datenbereich: Status (positiv = 00h / negativ = FFh) (1 Byte)
Globale Quittierung	85h (1000 0101)	• Siehe L4 Pakete • PaketNr. vom quittierten Paket • im Datenbereich: Checksumme des Datenbereichs des quittierten Pakets (2 Byte)

3.2.3 Vermittlung



Skizze Baummodell

Die Vermittlung im Protokoll stellt das eigentliche Kernproblem dar, mit dem wir uns beschäftigt haben. Normalerweise wird ein Netzwerk auf der untersten Ebene sternförmig organisiert. Dies führt dazu, dass die Kommunikation zwischen Clients fast immer über einen oder mehrere Server führt. Dies passiert im schlimmsten Fall auch dann, wenn sie direkt nebeneinander liegen. Veranschaulichen kann man sich das am Beispiel eines Baumes: Wenn man sich zwei Äste nimmt, deren Spitzen direkt nebeneinander liegen, müsste eine Ameise, zum Beispiel, unter Umständen bis zum zentralen Stamm krabbeln, um dann auf den anderen Ast zu gelangen.

Die Form der Vermittlung, die wir realisieren wollen, beruht darauf, dass alle am Datentransport beteiligten Stationen gleichberechtigt sind und es so keinen Unterschied zwischen Server und Client gibt.

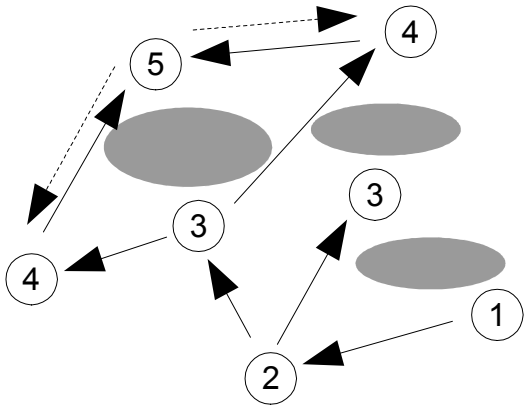
3.2.3.1 Heardlist

Jede Station führt eine sogenannte „Heardlist“ (Routingtabelle), in der unter anderem vermerkt ist, welcher Teilnehmer über welche Station mit welcher Anzahl von Weiterleitungen (Hops) erreichbar ist (Routing-Metrics). Dies entspricht in etwa der Routing-Tabellen eines Routers. Um diese Liste zu füllen, werden alle ankommende Pakete auf diese Daten hin ausgewertet. Wenn zwei Stationen, die in unserem Empfangsraum liegen, kommunizieren, werden alle Pakete ebenfalls mit ausgewertet, zum Beispiel, wo sie ursprünglich herkommen und zu welchen endgültigen Ziel sie gesendet werden sollen.

3.2.3.2 Pings

Als weitere Möglichkeit, die Heardlist zu füllen, stehen zwei verschiedene Pingtypen zur Verfügung. Eines wäre der lokale Suchping, der zu den direkten Nachbarstationen gesendet wird. Diese fragen darauf ihre Heardlist auf die gesuchte Station ab und teilen bei einem Treffer dies, mit der Anzahl der nötigen Hops, der suchenden Station mit. Falls diese Suche erfolglos bleibt, wird ein globaler Suchping ausgesandt. Dieser wird von jeder Station weitergeleitet und ist an die gesuchte Station gerichtet. Diese Suche breitet sich wie Wellen bis in die kleinste Ecke aus. Kommt dieser Suchping bei der gesuchten Station an, antwortet diese mit einem globalen Antwortping. Auch dieser wird dann von allen Stationen

weitergeleitet. Dies hat auch den Nebeneffekt, dass nun alle Stationen, die gerade online sind, wissen, wie diese Station zu erreichen ist. Der globale Antwortping wird ebenfalls gesendet, wenn eine neue Station eingeschaltet wird.



Beispiel eines sich bis in jeden Winkel weiterleitenden Signals

Diese Methode hat den großen Vorteil, dass ein Ping auch den letzten, kleinsten Winkel des Netzwerkes erreicht. Diese Weiterleitung führt aber auch zu einem großen Problem: Ein Ping, der von allen immer wieder weitergeleitet wird, verstopft irgendwann als „Geisterping“ das Netzwerk. Deshalb haben wir zwei Mechanismen eingebaut, die dies verhindern sollen:

Als 1. hat jeder Ping eine Nummer bekommen, die in der Heardlist vermerkt wird. Wenn also dieser Ping mit dieser Nummer ein zweites Mal bei der Station aufläuft, wird er ganz einfach ignoriert und nicht mehr weitergeleitet.

Als 2. wird innerhalb des Pingpaketes die Anzahl der Weiterleitungen gespeichert. Wird nun ein Ping öfter als ein gewisses Maximum weitergeleitet, wird auch er ignoriert.

Hiermit sind nun eigentlich die Voraussetzungen geschaffen, um eine erfolgreiche Kommunikation zwischen den Stationen zu gewährleisten, auch wenn der dazu nötige Kommunikationsweg sehr verschlungen ist.

3.2.3.3 Schwachstellen

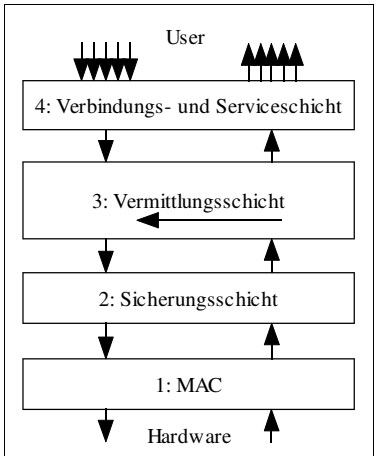
Leider sind bei der Erprobung einige Schwächen dieses Systems aufgefallen. Fehler treten um so häufiger auf, je höher die Anzahl der Clients pro Fläche ist. Zum einen wäre da ein Fehler in der Kanalzugriffskontrolle: Bei der Weiterleitung von Pings erhalten alle weiterleitenden Clients gleichzeitig das Senderecht, da ihre Anfrage erfolgt, während das Medium als frei deklariert ist. Somit überlagern sich die Weiterleitungen sehr stark und nur wenige erreichen ihre Ziele. Ein weiterer Fehler tritt auf, wenn ein Suchping auf sein Ziel stößt. Hierbei kollidiert die Antwort sehr wahrscheinlich mit verspäteten Suchpings, die über einen längeren Weg am Ziel ankommen. Generell entstehen diese Probleme bei einer hohen Netzwerkauslastung, welche bei einem globalen Ping entstehen kann. Diese sind um so schwerwiegender, da die Pings in unserem ursprünglichen Konzept nicht quittiert werden und somit der Übertragungsfehler nicht auffällt.

3.2.3.4 Lösungsmöglichkeiten

Um diese Probleme zu lösen, planen wir bis zum Landeswettbewerb die folgenden Bereiche nachzubessern. Der Backoff muss speziell bei Pings immer eingehalten werden, um Kollisionen unwahrscheinlicher zu machen. Um die übrigen Kollisionen, die nicht so einfach verhindert werden können, zu erkennen, werden die Pings, soweit es möglich ist, nicht nur per Broadcasting, sondern zusätzlich an alle im Umfeld bereits bekannten Clients, direkt versandt. Somit werden diese Pings von den einzelnen Clients quittiert und es fällt sofort auf, wenn es zu einer Kollision kam. Weiterhin können so nicht mehr vorhandene Clients erkannt und aus der Heardlist gestrichen werden, da es von ihnen dauerhaft keine Quittierung gibt.

3.2.4 Schichten

Wie oben schon beschrieben, haben wir uns am OSI-Schichtmodell orientiert. Schichten dienen dazu Funktionen zu bündeln (kapseln). Von außen sind bei den einzelnen Schichten nur wenige Schnittstellen zu erkennen. Um Aufgaben, die in den unteren Schichten zu erledigen sind, braucht sich eine höhere Schicht nicht mehr zu kümmern. Zum Beispiel hat die Schicht vier eine Funktion zur Messung der Laufzeiten. Hierfür braucht nur das Ziel angegeben werden, um alles andere kümmern sich die unteren Schichten. Es wird nur noch das Ergebnis, die Laufzeit, zurückgegeben.



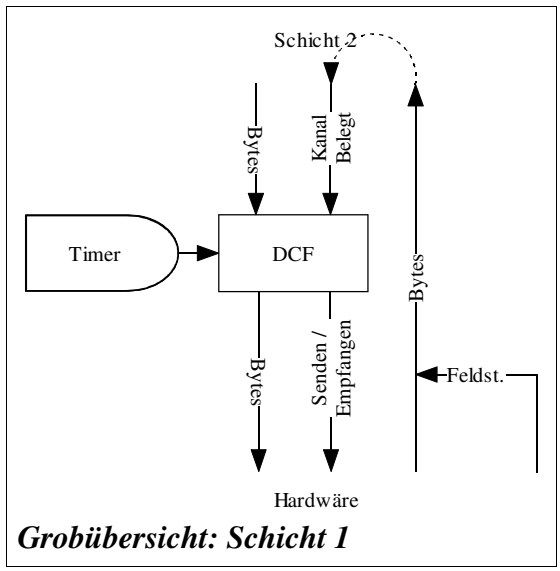
Schichten Grobübersicht

Für alle Schichten haben wir Flussdiagramme angefertigt. Diese hier einzubauen, würde jedoch den Umfang der Arbeit sprengen, da sie einen Umfang von 8 DIN A3 – Blättern umfassen. Bei der Vorstellung des Projektes können die Flussdiagramme eingesehen werden.

3.2.4.1 Schicht 1: MAC

Unter MAC versteht man die Mediumzugriffskontrolle (MediumAccessControl). Diese ist zuständig festzustellen, wann der Funkkanal frei ist, also die Wahrscheinlichkeit einer Kollision auf dem Funkkanal am Unwahrscheinlichsten ist. Das ist deswegen so wichtig, da man nach dem Sendebeginn im Funkverkehr nicht mehr feststellen kann, ob noch jemand anderes gleichzeitig sendet. So entstehen sehr große Verschwendungen an Bandbreite. Zusätzlich wird viel Zeit benötigt, eine Kollision zu entdecken. Um diesem Problem aus dem Weg zu gehen, haben wir einen Algorithmus, Distributed Coordination Function (DCF), aus dem WLAN (WirelessLocalAreaNetwork) entnommen und unseren Bedürfnissen angepasst, implementiert.

Hierzu wird nach dem Senden eines Pakets von allen, die senden wollen, eine Zufallswartezeit (Backoff) „ausgewürfelt“. Diese wird nun, solange keiner sendet, heruntergezählt. Wenn einer mit seiner Zeit bei Null angekommen ist, beginnt dieser zu senden, während die anderen ihre Zeit anhalten und sie beim Beenden des Sendevorgangs weiter zählen. Somit wird gewährleistet, dass alle Sendeversuche zeitlich gleichmäßig verteilt sind. Außerdem stürzen sich nicht alle nach Beenden eines Sendevorganges gleichzeitig auf den Funkkanal.

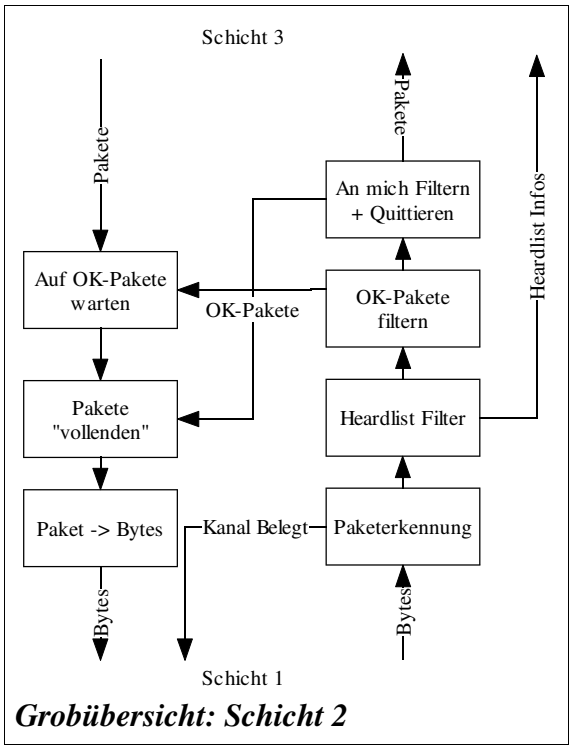


Grobübersicht: Schicht 1

Nachdem ein Client gesendet hat, sperrt er sich für eine gewisse Zeit selbst die Sendeversuche (ContentionWindow), um auch Andere zum Zug kommen zu lassen. Darüber hinaus gibt es ein weiteres Zeitfenster (DistributedCoordinationFunction InterFrameSpace, DIFS), das nach jedem einzelnen Sendevorgang besteht. In dessen ersten Hälfte besteht ein absolutes Sendeverbot, damit die Station, die gerade gesendet hat, wieder in den Empfangsmodus umschalten kann. Die zweite Hälfte ist dafür vorgesehen, Pakete mit einer hohen Priorität (Hi-Prio) zu versenden (z. B. OK-Pakete).

3.2.4.2 Schicht 2: Sicherungsschicht

Diese Schicht hat die Hauptaufgabe die ausgehenden Pakete in Bytes zu zerlegen und diese nacheinander zu senden. Nach jedem Sendeversuch eines Paketes aus der Schicht 3 wird auf eine Empfangsbestätigung der Gegenseite gewartet, dem OK-Paket. Wenn dieses in einer gewissen Zeitspanne nicht eintrifft, wird der Sendeversuch bis zu zweimal wiederholt.



Grobübersicht: Schicht 2

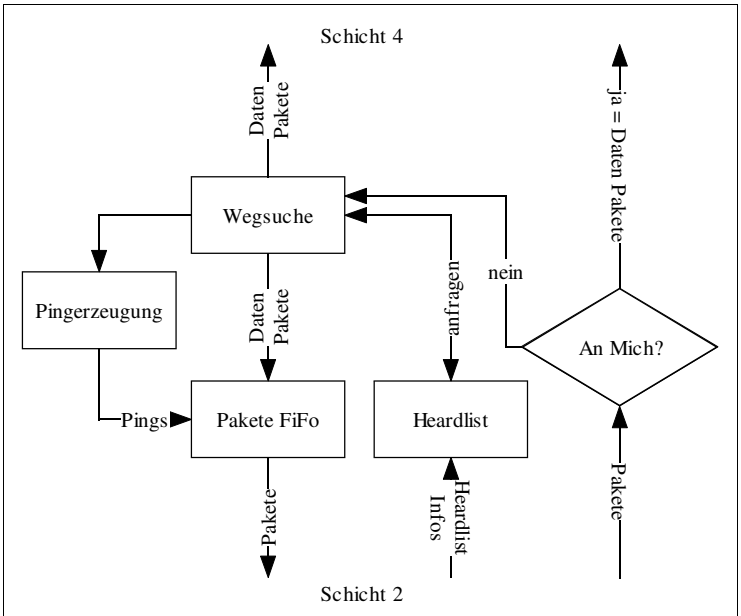
Auf der anderen Seite werden ankommende Bytes wieder in Pakete zusammengefügt. Sobald der Kopf der Pakete korrekt erkannt wird, wird dies der Schicht 1 mitgeteilt. Dies geschieht, damit Schicht 1 weiß, dass der Kanal in diesem

Moment belegt ist. Sobald die Pakete angekommen sind, wird auch dieses mitgeteilt. Die eingehenden Pakete werden zusätzlich alle heardlistrelevanten Einträge entnommen und an die Schicht 3 weitergeleitet. Als nächsten Schritt werden alle Pakete, die nicht an diese Station gerichtet sind, gelöscht. Weiterhin werden nun die OK-Pakete ausgefiltert und die dazugehörigen Quittierungen an die Schicht 3 gemeldet.

Eine weitere wichtige Aufgabe ist es, zu kontrollieren, ob die Pakete fehlerhaft sind oder nicht. Dies geschieht, indem z. B. Checksummen auf der ausgehenden Seite errechnet und auf der eingehenden Seite kontrolliert werden. Denn nur fehlerfreie Pakete werden an die Schicht 3 weitergeleitet.

3.2.4.3 Schicht 3: Vermittlungsschicht

Im Großen und Ganzen ist hier das realisiert, was unter dem Punkt 3.2.3 „Vermittlung“ beschrieben wird.



Grobübersicht: Schicht 3

Wenn ein Paket aus der Schicht 4 bzw. einer anderen Station zur Weiterleitung eintrifft, wird erst einmal in die Heardlist geschaut, ob der Empfänger dort verzeichnet ist. Wenn dieser Empfänger nicht in der Heardlist steht, wird eine Suche mit lokalen und globalen Pings, wie oben beschrieben, gestartet.

Eine weitere Aufgabe der Schicht ist es, die Datenbereiche der Pakete in ausgehender Richtung zu generieren und in eingehender Richtung zu zerlegen.

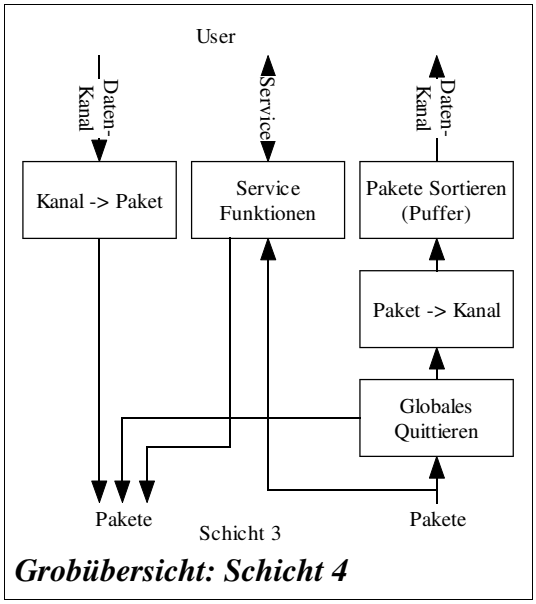
Letztendlich gibt die Schicht 4 in ausgehender Richtung nur noch die Anweisung, diese Daten, schon in Pakethäppchen zerteilt, an jene Adresse zu senden. Den Rest macht die Schicht 3, also Wegsuche, Weiterleitung, etc.

3.2.4.4 Schicht 4: Verbindungs- und Serviceschicht

Hier werden alle Serviceangebote zur Verfügung gestellt. Zum Einen werden Kanäle zu den einzelnen Stationen, mit denen häufig Daten ausgetauscht werden, simuliert. Dies dient dem Zweck, das nur noch der Kanal und nicht mehr die komplette Adresse angegeben werden

muss. Empfängerseitig wird damit klargestellt, dass Daten, die aus diesem Kanal kommen von einem bestimmten Absender stammen.

Weiterhin stellt diese Schicht die Pakete in die richtige Reihenfolge und kontrolliert, ob ein Paket im Datenstrom fehlt.



Grobübersicht: Schicht 4

Bei der Datenübertragung ist es möglich, dass eine Station für den ersten Teil der Übertragung einen langen Weg wählt, mittendrin jedoch feststellt, dass es noch einen Kürzeren gibt und den dann für den Rest benutzt. So ist es möglich, dass ein Paket schneller ist als ein anderes. Damit sichergestellt werden kann, dass Pakete vollständig beim richtigen Empfänger ankommen, ist die Schicht 4 des Empfängers für eine Quittierung der Pakete zuständig. Als letztes ist eine Funktion zur Messung der Laufzeit in dieser Schicht implementiert, der Laufzeitping.

4 Fazit

4.1 Zum Projekt

Unserer Auffassung nach haben wir ein System entwickelt, das genau auf die Bedürfnisse im Bereich der Telemetriedatenübertragung zugeschnitten ist. Es bildet eine Software- bzw. Protokollbasis für ein leicht einsetzbares Hardwaremodul. Zudem ist unser Projekt nicht nur auf einen stationären Einsatz beschränkt.

Eine tiefergehende mathematische Analyse der Leistungsfähigkeit unseres Projektes stellt allerdings ein eigenes Projekt für sich da. Da wären zum Beispiel die Fragen, ob sich Hauptverkehrsrouten ausbilden und bei welcher Belastung das Netzwerk zusammenbricht.

Wir mussten während der Erarbeitung des Projektes leider wieder feststellen, dass das Protokoll doch mehr Arbeit macht, als wir gedacht haben. Wir sind dieser Problematik allerdings mit einer guten Arbeitsmethode entgegnet, die wir uns beim Entwickeln des Protokolls angeeignet haben.

4.2 Zur Zukunft

Wie man an einer großen Anzahl von Publikationen zum Thema Ad-Hoc-Vernetzungen feststellen kann, ist das Interesse an diesem Zukunftsthema relativ groß. Auch die Industrie (z. B. Daimler-Chrysler) investiert, wie wir vor kurzem erfahren haben, in die Forschung auf diesem Gebiet.

Allerdings wird die Einführung dieser Systeme wohl noch auf sich warten lassen, da die Benutzer, zumindest in öffentlichen Netzen wie Handy-Netze, sicherlich ein Problem mit der Weiterleitung haben werden, da diese ihre Akkus belasten. Weiterhin besteht ein Sicherheitsproblem, da es nicht unmöglich ist, auf die weiterzuleitenden Daten zuzugreifen.

5 Die Hilfen

Diesen Absatz haben wir den Leuten gewidmet, ohne die dieses Projekt niemals in dieser Form zustande gekommen wäre. Allen vorweg natürlich der TGA, der Ausbildungsabteilung von Bosch – Blaupunkt, und damit Sebastians Abteilungsleiter, Herrn Königsdorff. Außerdem natürlich der Katholischen Hochschulgemeinde mit Thomas Harling, in der Marco seinen Zivildienst ableistet. Dort wurde öfters mal flexibel reagiert, wenn die Wörter „Jugend forscht“ fielen. Weiterhin wollen wir uns hier bei allen Freuden und Verwandten entschuldigen, die, obwohl sie nichts davon hatten, einiges an Stress und Arbeit mittragen mussten.

6 Verwendete Literatur

Bücher:

- Gerd Siegmund, Technik der Netze, R. v. Decker's Verlag Heidelberg 1996
- Werner Mansfeld, Satellitenortung und Navigation: Grundlagen und Anwendung globaler Satellitennavigationssysteme, Vieweg, 1998

Internet:

- OSI-7-Schichtenmodell: http://home.t-online.de/home/bkm-ieu151/Netzwerke/osi-7-schichtmodell.htm
- Rechnernetze – Wireless LAN http://einstein.kowalk.informatik.uni-oldenburg.de/rechnernetze/