

# Implementierung eines multithreaded TCP/IP Stacks für einen auf AMIDAR basierten Java Prozessor

Bachelorarbeit  
Robert Wiesner  
31. Mai 2017



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT



---

### **Erklärung gemäß § 22 Abs. 7 APB**

---

Hiermit erkläre ich gemäß § 22 Abs. 7 der Allgemeinen Prüfungsbestimmungen (APB) der Technischen Universität Darmstadt in der Fassung der 4. Novelle vom 18. Juli 2012, dass ich die Arbeit selbstständig verfasst und alle genutzten Quellen angegeben habe und bestätige die Übereinstimmung von schriftlicher und elektronischer Fassung.

Darmstadt, den 31. Mai 2017

Ort, Datum

Robert Wiesner

**Fachbereich Elektro- und Informationstechnik**

Institut für Datentechnik

Fachgebiet Rechnersysteme

Prüfer: Prof. Dr.-Ing. Christian Hochberger

Betreuer: Dipl.-Inform. Changgong Li

---

## Inhaltsverzeichnis

---

<b>1</b>	<b>Einleitung</b>	<b>3</b>
1.1	Motivation . . . . .	3
1.2	Aufgabenstellung . . . . .	3
<b>2</b>	<b>Grundlagen</b>	<b>4</b>
2.1	Netzwerk Schichtenmodell . . . . .	4
2.2	Ethernet . . . . .	4
2.2.1	Verfahren . . . . .	4
2.2.2	Ethernet Frame . . . . .	5
2.3	Internet Protocol Version 4 . . . . .	5
2.3.1	Paket Aufbau . . . . .	5
2.4	TCP . . . . .	5
2.5	DHCP . . . . .	5
2.6	AMIDAR . . . . .	5
<b>3</b>	<b>Implementierung</b>	<b>6</b>
3.1	Überblick . . . . .	6
3.2	TCP Stack . . . . .	6
3.3	IP Stack . . . . .	6
3.4	DHCP . . . . .	6
3.5	Stream Sockets . . . . .	6
<b>4</b>	<b>Evaluation</b>	<b>7</b>
4.1	Testaufbau . . . . .	7
4.2	Funktionen . . . . .	7
4.3	Performanz . . . . .	7
<b>5</b>	<b>Zusammenfassung</b>	<b>8</b>
5.1	Fazit . . . . .	8
5.2	Ausblick . . . . .	8

---

## **1 Einleitung**

---

### **1.1 Motivation**

---

Diese Arbeit basiert auf den im Fachgebiet Rechnersystem entwickelten AMIDAR Prozessor und der dafür angepassten Java API. Zum Zeitpunkt der Arbeit verfügt AMIDAR mit der dazugehörigen API über grundlegende Netzwerk Funktionalitäten. Dazu gehören die Unterstützung für die Protokolle Ethernet, ARP, begrenzt IPv4 und UDP. Mit UDP kann keine korrekte und verlustfreie Datenübertragung garantiert werden, welche für viele Netzwerkanwendungen vorausgesetzt wird. Im Rahmen dieses Projektes wird TCP Stack entwickelt, sowie der IP Stack erweitert um eine geordnete und verlustfreie Datenübertragung zu realisieren.

---

### **1.2 Aufgabenstellung**

---

---

## 2 Grundlagen

---

### 2.1 Netzwerk Schichtenmodell

---

Netzwerk Kommunikation zwischen Anwendungen wird üblicherweise als Schichtenmodell beschrieben. Die unterste Schicht stellt dabei das physical layer. Das beschreibt die Physische Übertragung von Daten. Darüber sorgt die Sicherungsschicht für eine funktionierende Verbindung zwischen Endgeräten und den Übertragungsmedium. Auf dieser Schicht wird zum Beispiel das Ethernet Protokoll eingesetzt, das die übertragenen Daten auf Fehler überprüft und im Zweifel verwirft. Darunter kommt die Vermittlungsschicht, in der die Endgeräte Adressiert werden und Routing und Datenflusskontrolle gesteuert werden. Ein wichtiges Protokoll dieser Schicht ist das IP Protokoll. 5blabla

Bei einer Datenübertragung von einer Anwendung zu einer auf einen anderen Endgerät laufenden werden die Daten in durch die Schichten nach unten gereicht, wobei in jeder Schicht ein neuer Header erzeugt wird, der für die jeweilige Schicht wichtige Informationen enthält. Zum Beispiel werden die Daten von der Anwendung mit betriebsystemsabhängigen Systemaufrufen an den TCPStack übergeben. Dieser erzeugt ein TCP Paket, das außer den Daten einen Header enthält, welcher Information bereitstellt, die unter anderen für das richtige Zusammensetzen der einzelnen Datenpakete beim Empfänger, als auch für die Zuordnung der übertragen Daten zu der jeweiligen Anwendung benötigt werden. Bei der anschließenden Erzeugung des IP Pakets bilden das TCP Paket bestehend aus Daten und TCP-Header die zu übertragenden Daten. Der IP Header enthält unter anderen die IP-Adressen des Ziel und Quell Geräts. Das IP Paket bleibt im Normalfall unverändert, bis das Zielgerät erreicht ist. An der nächst unteren Ebene steht das Ethernet Datagramm. Es enthält neben dem IP Paket die Physischen Adressen von des Quell Endgeräts und der nächsten Zwischenstation auf dem Weg zum Ziel. Bei Zwischenstation wird anhand der der Daten des IP Pakets der nächste Wegpunkt ermittelt und ein neues Datagramm erzeugt.

Wenn ein Datagramm das Ziel erreicht wird das IP Paket extrahiert und daraus das TCP Paket. Anhand der Port Nummer kann das TCP Paket der Anwendung zugeordnet werden.

---

### 2.2 Ethernet

---

Ethernet nach der IEEE Norm 802.3 ist seit den 90ern der am weitesten verbreitete Standard für Lokale Netzwerke und beschreibt sowohl die Bitübertragungs als auch die Sicherungsschicht.

---

#### 2.2.1 Verfahren

---

Um zu ermöglichen das mehrere Endgeräte auf den selben Physischen Medium kommunizieren können wurde früher ein Zeitmultiplexverfahren eingesetzt, das durch den CSMA/CD Algo-

---

rhythmus gesteuert wird. Wenn eine Stelle Daten zum senden bereithält, wartet diese bis das Medium ungenutzt ist und fängt dann an die Daten zu übertragen. Wenn 2 Stellen gleichzeitig beginnen zu senden wechseln beide auf ein SStörung-ErkantSSignalmuster und beenden die Übertragung. Nach einer zufällig langen Pause wird jeweils ein erneuter Übertragungsversuch gestartet.

Mittlerweile werden Kollisionen durch die Einführung von Switches verhindert. In diesen können Ethernet Pakete zwischengespeichert werden bis diese gesendet werden können. Dadurch wird eine Vollduplex Übertragung zwischen Switches und anderen Endgeräten ermöglicht. Es kann jedoch vorkommen, das Switches bei zu großen Datenaufkommen überlastet werden weswegen die Ethernet Flow Control Datenpakete verwerfen kann. Deswegen ist es wichtig, das Protokolle auf den darüber liegenden Schichten verworfene Datenpakete erkennen und erneut senden können um eine zuverlässige Datenübertragung zu gewährleisten.

---

### 2.2.2 Ethernet Frame

---

Ein Ethernet Paket beginnt mit einer sieben Bit langen Präambel die aus einer alternierenden Folge von Einsen und Nullen besteht. Diese wird für die Synchronisation der Verbindung benötigt und ermöglicht es die Folgen Daten von Hintergrundrauschen zu unterscheiden. Unterbrochen wird die Präambel durch das auf Eins gesetzte SStart of Frame"Bit was mit den letzten Bit der Präambel 2 aufeinander Folgende Einsen ergibt. Der eigentliche Ethernet Frame beginnt mit der aus Sechs Byte bestehenden Ziel MAC-Adresse gefolgt von der Quell MAC-Adresse. Dazu kommen 2 Byte die den Typ des darüber liegenden Protokolls angeben. Zum Beispiel 0x0800 gibt IPv4 an. Dahinter kommen 46-1500 Bytes an Daten, gefolgt von 4 Bytes Frame Check Sequence. Diese besteht aus einer CRC Checksumme.

---

## 2.3 Internet Protocol Version 4

---

Das IP Protokoll ist das für die Datenübertragung wichtigste Protokoll, auf der Vermittlungsschicht. Es wurde Entwickelt um eine paketvermittelte Kommunikation über mehrere Computernetzwerke hinweg zu ermöglichen. Quellen und Ziele der Übertragungen werden jeweils als Adressen mit fester 32 Bit Länge angegeben. Es gibt keine Mechanismen für Zuverlässige Übertragung, Flusskontrolle und Sequenzierung. Es gibt jedoch Möglichkeiten zur Paketfragmentierung, falls Datenpakete die Maximale Segment Größe für Pakete der darunter liegenden Schicht überschreiten sollten.

---

### 2.3.1 Paket Aufbau

---

---

## 2.4 TCP

---

---

## 2.5 DHCP

---

---

## 2.6 AMIDAR

---

---

## **3 Implementierung**

---

### **3.1 Überblick**

---

### **3.2 TCP Stack**

---

### **3.3 IP Stack**

---

### **3.4 DHCP**

---

### **3.5 Stream Sockets**

---

---

## **4 Evaluation**

---

### **4.1 Testaufbau**

---

### **4.2 Funktionen**

---

### **4.3 Performanz**

---



---

## **5 Zusammenfassung**

---

### **5.1 Fazit**

---

### **5.2 Ausblick**

---

---

## Abkürzungsverzeichnis

---

**CLB** Configurable Logic Block

**CPU** central processing unit

**ELF** Executable and Linking Format

**FPGA** Field Programmable Gate Array

**MC** Microcontroller

**CPU** central processing unit

**NCD** Native Circuit Description Format

**SoC** System on Chip

**Sph** SpartanMC Hex Dateiformat

**XDL** Xilinx Design Language Format

---

## Abbildungsverzeichnis

---

---

## Literaturverzeichnis

---

- [BUG] *BUG Xilinx XDL Programm*. <http://forums.xilinx.com/t5/Spartan-Family-FPGAs/A-bug-for-Spartan-6/td-p/271026>
- [ELF] *Executable and linkable Format*. <http://www.linux-kernel.de/appendix/ap05.pdf>
- [GNU] *GNU - make*. <http://www.gnu.org/software/make/manual/make.html>
- [OSE] *Open Source ELF-Library*. <https://www.hpi.uni-potsdam.de/hirschfeld/trac/SqueakMaxine/browser/com.oracle.max.elf?rev=1d81ff0c9a2a8a101fe33f4ac1495decc6147517#src/com/oracle/max/elf>
- [UCS] *XDL Use Case Senarios*. [http://www.cs.indiana.edu/hmg/le/project-home/xilinx/ise\\_5.2/help/data/xdl/xdl-ucs-ext.html](http://www.cs.indiana.edu/hmg/le/project-home/xilinx/ise_5.2/help/data/xdl/xdl-ucs-ext.html)
- [XDL] *Xilinx Design Language*. [http://www.cs.indiana.edu/hmg/le/project-home/xilinx/ise\\_5.2/help/data/xdl/xdl.html](http://www.cs.indiana.edu/hmg/le/project-home/xilinx/ise_5.2/help/data/xdl/xdl.html)