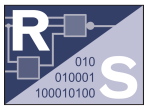
Implementierung eines multithreaded TCP/IP Stacks für einen auf AMIDAR basierten Java Prozessor

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

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 Datenpakte 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-

rithmus 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-ErkanntSSignalmuster 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 (IPv4)

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, weswegen das darüber liegen Protokoll dies sicher stellen muss. 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

Version: Gibt in an welche Version des IP Protokolls verwendet wird. (4Bit)

IHL: Steht für Internet Header Length und gibt an wie 32Bit Wörter von dem IP-Header belegt werden.

ToS: Type of Service beinhaltet abstrakte Parameter zur Bestimmung der Qualität des gewünschten Services. Dabei geben die Bits Null bis Zwei die Priorität der Daten an. Die Bits Drei, Vier und

Fünf stehen für niedrige Latenz, hohen Durchsatz und hohe Zuverlässigkeit. Die letztgenannten Parameter werden von Netwerkgeräten von unterschiedlich interpretiert, in dem meisten ruft bessere Performance für einen der Parameter eine Verschlechterung bei einen anderen herbei.
Paketlänge: Gesamtlänge eines Pakets in Bytes einschließlich des Headers. Die 16Bit ergeben eine theoretische Gesamtlänge von 65.535 Bytes, was für viele Netzwerke jedoch nicht geeignet ist. Als Mindestgröße die alle Hosts unterstützen müssen wurde 576 Bytes festgelegt. Das ermöglicht es 512 Byte Daten und 64 Byte Header in einen Paket zu übertragen. Da der IP Header selber nur 20 Byte benötigt bleibt noch ein Puffer von 44 Byte für den Header des darüber liegenden Protokolls.

Kennung: Ein Identifikationswert, der benötigt wird um fragmentierte Pakete wieder zusammen zu setzen. (16Bit)

Flags: 3 Bits von denen das erste reserviert ist und dauerhaft auf Null gesetzt wird. Das zweite Bit gibt an, ob das Paket fragmentiert werden darf. Das letzte Bit wird gesetzt, wenn nach dem Paket noch weiter Fragmente folgen.

Fragment-Offset: Besteht aus 13 Bit und gibt an, an welche Stelle des Datagramms die Daten dieses Fragments gehören.

TTL (Time-to-live): Gibt die maximale Zeit in Sekunden an, die ein Paket im Netzwerk unterwegs sein darf. Sobald die TTL den Wert Null erreicht muss das Paket gelöscht werden. Da der Wert bei jeder Zwischenstation, unabhängig von der eigentlichen Verarbeitungszeit ebenfalls um eins Reduziert werden muss ist die übliche Lebensdauer eines Pakets deutlich kürzer als in der TTL angegeben.

Protokoll: Gibt an welches Protokoll im nächst höheren Level verwendet wird.

Header Checksumme: Checksumme nur über den Header. Da sich der Header beim Weg Zwischenstationen verändern kann, zum Beispiel wegen der Time to Life , muss die Checksumme nach jeder Verarbeitung des Headers an den Zwischenstationen neu berechnet werden.

Quell-IP-Adresse: IP Adresse des Hosts, der das Paket ursprünglich versendet hat.(4 Byte)

Ziel-IP-Adresse:IP Adresse des Zielendgeräts. (4 Byte)

Optionen/Füllbits

2.3.2 Adressierung

Um mehrere Netzwerke innerhalb eines großen zu ermöglichen, werden IP Adressen interpretiert, in dem eine bestimmte Menge der höherwertigen Bits das Netzwerk spezifiziert und die restlichen Bits den genauen Hosts des gewählten Netzwerks adressieren. Dabei soll Flexibili-

tät bei der Unterteilung von kleineren Teilnetzwerken ermöglicht werden. Daher wurde das Adressfeld entsprechend in bestimmte Klassen unterteilt. Es wurden 3 Klassen A,B,C angelegt. Die Klassen unterscheiden sich jeweils durch die Aufteilung des Adressbereichs zwischen Netzwerk Adresse und Host Adresse innerhalb des Netzwerks. Die ersten 1-3 Bits der Adresse geben jeweils Ausschluss darüber zu welcher Netzwerkklasse die jeweilige IP-Adresse gehört. Um innerhalb dieser starren Netzwerke feiner Aufgeteilte Subnetze zu erzeugen kann die Subnetmask genutzt werden. Die Subnetmask ist ebenfalls eine 4 Byte Zahl, deren niederwertige Bits auf Null gesetzt werden und damit den Variablen Teil innerhalb des subnetz angeben. Der Rest der Maske wird dabei auf Eins gesetzt.

2.3.3 Fragmentierung

Paketfragmentierung wird nötig, wenn ein Paket aus einen Netzwerk kommt das eine große maximale Segmentgröße erlaubt und durch eines geleitet wird, das nur eine kleinere Segmentgröße erlaubt. Dabei können die Pakete in eine theoretisch nahezu endlose Anzahl von kleinen Paketen zerlegt werden. Dabei müssen die Datenblocks alle Fragmente bis auf das letzte ein vielfaches von 64 Byte an Daten beinhalten. Damit fragmentierte Paket richtige zusammen gesetzt werden können, haben alle Fragmente, eines Pakets, die selbe Identifikationsnummer. Für das Zusammensetzen des Datenblock wird der Fragmentoffset benötigt der angibt, an welcher Stelle des ursprünglichen Datenblocks die Daten des Fragments stehen. Wobei das fragmented-flag angibt, ob noch weitere Fragmente folgen, oder ob dies das letzte Teil ist. Sobald alle Fragmente eines Pakets beim Ziel eingetroffen sind, kann das ursprüngliche Paket wiederhergestellt werden.

2.4 Transmission Control Protocol (TCP)

Da die Protokolle der unteren Schichten, IP und Ethernet keine fehlerfreie und Verlustlose Übertragung der Daten garantieren können, wird ein Protokoll auf der Transportschicht benötigt, das eine zuverlässige Übertragung von Daten zwischen Anwendungen auf entfernten Hosts sicherstellen kann, auch wenn auf jeden Hosts eine große Anzahl an Anwendungen läuft, die TCP verwenden. Für diesen Zweck wurde das TCP-Protokoll erschaffen. Um dabei die Datenpakete jeweils der richtigen Anwendung zuordnen zu können werden Port-nummern verwendet.
Es handelt sich bei TCP um ein verbindungsorientiertes Protokoll, das bedeutet, dass es zu beginn einer Übertragung einen klar definierten Verbindungsaufbau gibt. Nachdem die Verbindung etabliert es können Daten vollduplex in beide Richtungen übertragen werden. Wobei durch Sequenznummern und Acknowledge Nummern die richtige Reihenfolge und Vollständigkeit der Datenpakete sichergestellt wird. Pakete deren Erhalt nicht bestätigt wurde werden automatisch neu übertragen. Desweiteren verfügt TCP über Mechanismen um eine Überlast auf dem Übertragungsweg rechtzeitig zu erkennen und zu beheben.

2.4.1 Paketaufbau

Quell Port (16Bit): Gibt die Portnummer der Anwendung auf Senderseite an
Ziel Port (16Bit): Gibt die Portnummer der Anwendung auf Empfängerseite an
Sequence Number (32Bit):
Acknowledge Number (32Bit):
Data Offset (4Bit):Anzahl von 32Bit Wörtern aus denen der Header besteht.
Reserved (6Bit): reserviert für zukünftige Nutzung
Steuerungsbits (6Bit): 6 Flags die für die Ablauf Steuerung der Übertragung genutzt werden:
URG: Urgent Pointer, wird in moderner Software nicht mehr genutzt und wird von vielen Implementierungen ignoriert.
ACK: Acknowledgment gibt an, ob das Paket eine gültige Bestätigung für empfangene Pakete enthält.
PSH: Push, ist dieses Flag gesetzt, werden die empfangenen Daten sofort an die Hostsoftware weitergereicht. RST: Resettet die Verbindung. Wird im Fehlerfall gesendet und bricht die Verbindung ab. FIN: Signalisiert das ende der Verbindung, wenn es keine zu übertragenen Daten gibt.

Window (16Bits): Gibt die Größe des Emfpangswindows des Absenders an. Gilt für den Empfänger als obere Grenze des Congestion Windows
Checksum (16Bits): Für die Berechnung der Checksumme über das TCP Paket wird vorher ein Pseudoheader bestehend aus den Quell- und Ziel IP-Adresse, des IP-Codes für das verwendete Protokoll und die Gesamtlänge des eigentlichen TCP Pakets. Die Checksumme wird daraufhin über den Pseudeheader, den Header und die Payload berechnet. Dafür werden diese in 16 Bit große Blöcke aufgeteilt und im einer Komplement die Summe über diese gebildet.
UrgentPointer (16Bits): Gibt den Urgentpointer als Offset zu der Sequenznummer an. Wird nur interpretiert, wenn das URG Flag gesetzt ist.
Options: (Variabel): Optionale Informationen, die von TCP Implementierungen unterstützt werden können um Sicherheit und Performance zu verbessern. Optionen bestehen entweder nur aus einen Optionsbyte oder haben zusätzlich ein weiteres Byte das die Länge der jeweiligen Op-

tionen angibt. Padding: Falls der TCP Header mit den Optionen einen teilweise genutzten 32Bit Block hat, wird dieser mit Nullen aufgefüllt.

2.4.2 Zustände

TCP ist im Gegensatz zu den bisher genannt Protokollen Zustandsorientiert, was bedeutet, das es ja nach Zustand anders auf Nutzeraktionen und ankommende Pakete reagiert. Zu den Zuständen gehören : Closed : Das ist der Start zu stand einer TCP Instanz. Die Verbindung ist geschlossen. Usercalls außer Öpen"werden mit Fehlermeldungen quittiert und ankommende Pakete werden verworfen und mit einen Reset-Paket beantwortet.
SYN-sent: Nachdem eine Verbindung initiiert wurde, wird auf eine Antwort des "remote Hosts"gewartet.
SYN-received : Nachdem ein SYN Pakte erhalten und ein SYN-ACK gesendet wurde wird auf das ACK gewartet, um den 3-Wege-Handschlag abzuschließen.
ESTABLISHED: Nach dem der Verbindungsaufbau erfolgreich abgeschlossen wurde befinden sich beide Hosts im ESTABLISHED Zustand, in dem die eine Vollduplex Komminikation möglich ist.
FIN-WAIT-1: Wenn ein Verbindungsabbau initiert wurde wird ein FIN Paket gesendet und in den Zustand FIN-WAIT-1 gewechselt und auf die Bestätigung des Erhalts gewartet.
FIN-WAIT-2: In dem Fall das die Gegenstelle noch Daten zu übertragen hat, bleibt die Verbindung einseitig offen um die letzten Pakete zu empfangen.
CLOSING:
TIME-WAIT:

Zu den Zuständen kommen noch eine Reihe von Ereignissen auf, auf die, abhängig von den jeweiligen Zuständen entsprechend reagiert werden muss. Dazu gehören neben eintreffenden Paketen und Timeouts die Usercalls.
Active OPEN: Öffnet einen Port und initiiert den Verbindungsaufbau zu einen Remote Host.
Passive OPEN: Öffnet einen Port ohne eine Verbindung zu initiieren und wartet auf einen Verbindungsaufbau,
SEND: Fügt dem Sendepuffer Daten hinzu und sendet gegebenenfalls ein Datenpaket.
RECEIVE: Überprüft, ob genug Daten vorhanden sind und gibt diese an die Anwendung zurück.
CLOSE: Initiiert den Abbau der Verbindung, wenn es keine Daten mehr zu übertragen gibt.
ABORT: Bricht die Verbindung ab. In den meisten Zuständen wird in dem Fall ein Reset Paket gesendet.

2.4.3 Sequenznummern

Ein wichtiges Grundprinzip von TCP das jedes Byte an Daten eine eigene Sequenznummer zugewiesen werden kann, dadurch ist es möglich, das jedes Byte einzeln bestätigt werden kann. Der in TCP dazu verwendete Mechanismus arbeitet kumulativ. Das bedeutet das wenn eine Sequenznummer bestätigt wird, alle Sequenznummern kleiner als die Bestätigungsnummer als angekommen gelten. Dadurch wird es einfach den Verlust einzelner Pakete zu erkenne und die-

se nochmal zu senden. Das erste Byte eines Datenpaketes entspricht dabei der Sequenznummer des Pakets.

2.4.4 Verbindungsaufbau

2.4.5 Datenübertragung

2.4.6 Überlastkontrolle

2.4.7 Verbindungsabbau

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

[XDL] *Xilinx Design Language*. http://www.cs.indiana.edu/hmg/le/project-home/xilinx/ise_5.2/help/data/xdl/xdl.html