Longitudinal study of user-space packet processing frameworks in academia

Längsschnittstudie von User-Space Paketverarbeitung Frameworks in der Wissenschaft

Maik Luu Bach
Betreuer: Paul Emmerich
Seminar Innovative Internet-Technologien und Mobilkommunikation SS2017
Lehrstuhl für Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: maik.luu-bach@tum.de

KURZFASSUNG

Es existieren heutzutage viele Packet Processing Frameworks. Sie erfüllen größtensteils denselben Zweck und unterscheiden sich in diversen Eigenschaften. Gegenwärtig ist es möglich Fast Packet Processing mit kostengünstige gebrauchsübliche Hardware zu betreiben. Einige Packet Processing Frameworks stehen dabei heraus. Als Ansatz, um herausfinden welches das meist genutzte Framework ist, wurden wichtige Konferenzen der Datenkommunikation in Betracht genommen.
In dieser Arbeit wird dargelegt, warum gerade Frameworks, wie netmap oder DPDK auch in Facharbeiten eine große Bedeutung haben und wie sie in den letzten Jahren an Popularität gewonnen haben.

Schlüsselworte

packet processing, DPDK, netmap, Java, PDFBox

1. EINLEITUNG

Das Internet produziert immer neue Daten, Tag für Tag. Schätzungen von 2015 besagen, dass sich das Volumen der Daten bis 2025 verzehnfachen wird auf 163 Zettabyte. Das ist eine Wachstumsrate von circa 30% pro Jahr [1]. Diese Datenmenge muss weitertransportiert werden, zum Beispiel innerhalb eines Unternehmens. Bessere und zugleich auch alternative Technologien werden entwickelt, um neue Ansätze Datenpakete zu bearbeiten und zu transferieren. Commodity-Hardware steht im Vordergrund der neusten Forschungen, sodass jeder mit seiner vorhandenen Hardware und gegebener Software effizientes Packet Processing betreiben kann [2].

Frameworks, wie z.B. DPDK [3] und netmap [4] entstehen und Forscher aus aller Welt versuchen eine gute Mischung verschiedenster Frameworks zu finden. Laut Broy ([5]) ist eine Framework „ein halbfertiges Softwaresystem, das noch vervollständigt werden muss“. Zudem liefern die meisten Frameworks schon die Treiber für eine ausführbare Implementation.

In Abschnitt 2 werden die Grundlagen des User Space präsentiert. Im dritten Abschnitt der Arbeit werden die verschiedenen Arten und Zuordnung der Frameworks vorgestellt. Ein kurzer Vergleich wird im Abschnitt 4 dargelegt. Abschnitt 5 werden die Organisationen und dazugehörigen Konferenzen nacheinander genannt. Die Vorgehensweise der Beschaffung der Publikationen wird im sechsten Abschnitt beschrieben. Das anschließende Ergebnisse mit Zusammenfassung stehen am Ende dieser Seminararbeit.

2. USER SPACE

Ein Systemspeicher in Linux ist in 2 Kategorien einzuordnen, in Kernel Space und User Space. Der Kernel Space definiert den Speicher, wo der Betriebssystemkern seinen Code speichert und ausführt wie Treiber und Dateisystem [6].

Das User Space wiederum beschreibt den Teil des Systemspeichers, wo der Nutzer seine Prozesse exekutiert [7]. Darunter zählen nicht nur die Programme des Benutzers, sondern auch die enthaltenen Bibliotheken mit deren Methoden. Die Abbildung 1 veranschaulicht dies vereinfacht dar [8].

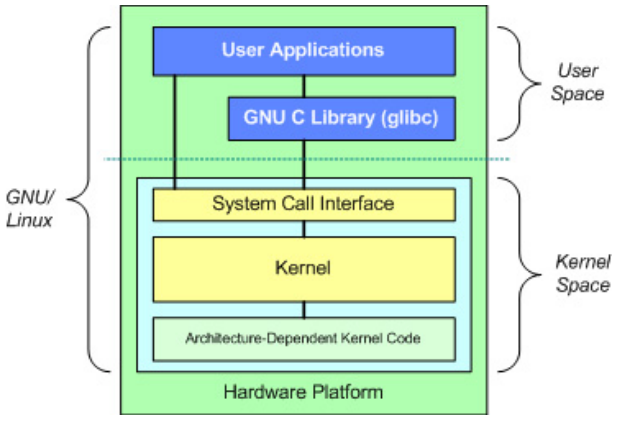

Abbildung 1: Grundlegende Architektur von GNU/-Linux

3. ARTEN VON PACKET PROCESSING FRAMEWORKS

Grundsätzlich kann man heutzutage die Packet Processing Frameworks in 2 Grundkonzepte aufteilen, welche die die CPU als zentrale Einheit zur Übertragung der Pakete nutzt,

doi: 10.2313/NET-2018-11-1_07

und andererseits den Ansatz die GPU in den Mittelpunkt zu stellen.

3.1 Techniken der Paketverarbeitung

Bevor wir zu den einzelnen Frameworks kommen, sind Erläuterungen notwendig zum besseren Verständnis der Thematik. Dazu beschreibe ich in den folgenden Absätzen die „Zero-Copy"-Technik und die Stapelverarbeitung (im Englischen „batch-processing") genannt.

3.2 Zero-Copy Technik

Ein Kernproblem der Paketverarbeitung ist unteranderem der begrenzte Speicherplatz. Zwischen Kernel und User Space wird so unnötig Speicherplatz verbraucht. Um dies zu vermeiden, existiert es die „Zero-Copy"-Technik. Diese Technik ermöglicht es, anhand einer Sammlung von Techniken, die Anzahl der Kopien zwischen Kernel und Geräten oder Kernel und User Space zu verringern. Dadurch ist eine Implementation in Umgebungen möglich, die wenig Arbeitsspeicher besitzen und so kostengünstige Lösung erlauben [12].

3.3 Stapelverarbeitung

Die Stapelverarbeitung ist eine Technik der Datenverarbeitung. Man kann sich einen Stapel vorstellen, wo alle Aufgaben in einem Stapel gesammelt werden und von oben nach unten abgearbeitet werden. Die Verarbeitung verläuft meistens automatisch und sequenziell [9].

3.4 CPU

Durch die Nutzung der Frameworks kann im Prinzip auch ein einfacher Bürocomputer als Router fungieren. Die Vorteile bestehen aus ihrer hohen Flexibilität, weil hauptsächlich nur Software benötigt wird, die schon in den meisten Fällen „open-source" ist. Der Gebrauch von standardardisierter Hardware ermöglicht eine preiswerte Einsetzung der Software-Router. Leider bietet diese Perspektive keine dauerhafte kommerzielle Lösung, da schlicht die Leistung dazu fehlt und spezielle Hardware-Router den Zweck besser erfüllen [13].

3.4.1 Data Plane Development Kit (DPDK)

DPDK ist ein open-source Linux Foundation Projekt mit dem Ziel eine Zusammenstellung von Treibern und Bibliotheken für das Fast Packet Processing zu liefern, unabhängig welche CPU-Architektur der Benutzer besitzt. Außerdem unterstützt DPDK zum Beispiel auch eine Multicore Framework, sodass eine maximal effiziente Weise mit Algorithmen geschaffen wird für das Packet Processing [3]. DPDK kann mit der Zero-Copy Technik angewendet werden und spart damit die Anzahl der Kopien im Arbeitsspeicher [12].

3.4.2 netmap

Die Framework netmap wird genutzt für High-Speed I/O und wird implementiert mit der mitgelieferten VALE Software als Single-Kernel Modul. Durch netmap können auf Netzwerkkarten, Host-Stacks, virtuelle Ports und „Netmap-Pipes" eine Bandbreite von über 14 Mpps erreicht werden ohne teure Hardware einzusetzen [4]. Netmap kann auch mit der Zero-Copy Technik angewendet werden [12].

3.4.3 PF_RING

PF_RING wird, wie die anderen beiden Frameworks zuvor, mit Zero-Copy Technik umgesetzt. Mithilfe von hunderten zusätzlichen Filtern und Paketanalyse versucht PF_RING die Paketerfassung zu optimieren. Es existieren verschiedene Typen von PF_RING zum Beispiel DNA (Direct NIC Access) und ZC (Zero-Copy) [17].

3.4.4 Snabb

L. Gorrie entwickelt seit 2012 die High-Performance Framework Snabb [10]. Diese Framework läuft auf Linux und x86-64 Architekturen. Zur Umsetzung des Systems nutzt man die Programmiersprache Lua und einen Compiler namens „LuaJIT", was es kompatibel macht mit C. Hierdurch besitzt das Framework einen „kernel-bypass" und kann außerhalb der User-Space, sogar im Kernal-Space ausgeführt werden [11].

3.4.5 PFQ

Ein Linux-basierte Framework ist PFQ, mit dem Gedanke wie DPDK, effizient Daten mit Skripten und Algorithmen zu beschleunigen. Die Implementierung von PFQ wird eigens mit der Programmiersprache PFQ-Lang umgesetzt, welche inspiriert von Haskell. Die Ergebnisse von PFQ sind sehr hardware-abhängig [16].

3.5 GPU

Der Vorteil gegenüber den CPU-betriebenen Frameworks ist die I/O Bandbreite. Der aktuelle PCI-Express 4.0 Standard kann auf einen 16-Lanes Steckplatz bis zu 16 GB/s übertragen. Schon in den nächsten Jahren wird ein Standard veröffentlicht, deren Datenrate über 32 GB/s beträgt und neue Möglichkeiten sich dadurch erschließen in der Thematik Performance. Nachteil dieses Ansatzes ist zum Teil die Umsetzung. Mainboards haben eine begrenzte Anzahl von PCI-Express-Slots und die Netzwerkkarte sollte dieselbe Bandbreite haben, um einen Bottleneck zu vermeiden [13].

3.5.1 PacketShader I/O

PacketShader unterscheidet sich vom Ansatz der anderen Frameworks, denn sie nutzt die GPU des Computer für das Packet Processing. Das zeigt das Grafikkarten nicht nur für Spiele und für das Kryptominen gut sind, sondern auch für Paketverarbeitung [13]. Die CPU wird nicht mehr als Bottleneck der Performance angesehen. Die Kosteneffizienz und das Potenzial einer parallelen Datenverarbeitung sind einige Beispiele der Vorteile von PSIO. Zur Zeit ist die einzige Begrenzung 40 GB/s, verantwortlich dafür ist die Einschränkung der Kapazitäten des Dualen I/O Hubs [15].

4. VERGLEICH

Jede Paketverarbeitungs-Framework hat seine Vor- und Nachteile. In diesem Abschnitt beschreibt die Abbildung 2 vereinfacht den Vergleich zwischen den Packet Processing Frameworks in diversen und prägnanten Eigenschaften.

5. BETRACHTETE KONFERENZEN

5.1 ACM

Mit den Gedanken „to advancing the art, science, engineering, and application of information technology" wurde bei einer Sitzung an der Columbia Universität im Jahre 1947 die Association for Computing Machinery gegründet. Sie ist

doi: 10.2313/NET-2018-11-1_07

	netmap	PF_RING	DPDK	pfq	Snabb	PSIO
Memory preallocated,reused	yes	yes	yes	yes	yes	yes
Parallel direct paths						yes
Memory mapping	yes	yes	yes	yes	yes	yes
Zero-copy	yes	yes	yes			
Batch processing	yes			yes		yes
GPU processing						yes

Abbildung 2: Vergleich der Frameworks (abgewandelt von [12])

damit die älteste wissenschaftliche Gesellschaft für die Informatik [18]. Unterteilt ist die Organisation in 34 Special Interest Groups [19].

5.1.1 SIGCOMM

Einer der Untergruppierungen der ACM ist die SIGCOMM (Special Interest Group on Data Communications) spezialisiert in Datenkommunikation und Computernetzwerke. Mit eine allgemein geringe Akzeptanzrate von 13% spricht für das Niveau der alljährlichen Konferenz [20].

5.2 IEEE

Das „Institute of Electrical and Electronics Engineers“ ist der weltweit größter Berufsverband, mit Mitgliederzahlen über 420.000, von Ingenieuren aus den technischen Bereich. Die IEEE publiziert jedes Jahr rund ein Drittel der Literatur aus diesem Segment [21]. Außerdem setzt sich die Zweckgemeinschaft für Standardisierungen ein.

5.2.1 ANCS

Die ANCS wird jedes Jahr mit Kooperation der ACM veranstaltet. Ziel des Forums ist die Forschung der Synergie von Algorithmen und Netzarchitekturen in der Theorie, sowie auch in der Praxis.

5.3 USENIX

Die USENIX ist eine Vereinigung von Entwicklern seit 1975. Sie nehmen sich als Ziel technische Innovationen zu fördern und verbreiten Arbeiten mit einem praktischen Hintergrund. Zudem bietet die Organisation ein neutrales Forum für Diskutionen [23].

5.3.1 ATC

Die USENIX Annual Technical Conference besprechen Forscher der Informatik über Themen wie Virtualisierung, Cloud-Computing und unter anderem auch Netzmanagement [24].

5.3.2 NSDI

Sponsiert wird die NSDI von USENIX mit Kooperation der ACM Gruppen SIGCOMM und SIGOPS. Hauptthemen sind Design und Implementation von Netwerksystemen [25].

5.3.3 OSDI

Alle 2 Jahre findet die OSDI Konferenz statt, wo die Hauptthematik Betriebssysteme stehen. Dabei betrachten die Spezialisten Theorie, sowie auch die Praxis [26].

6. VORGEHENSWEISE

In diesem Kapitel wird die Herangehensweise erläutert. Um die Beschaffung der Dokumente zu vereinfachen und zu beschleunigen, wurden viele Bibliotheken und kleine Hilftools wie Add-ons für diverse Browser. DownThemAll gibt einem die Möglichkeit Dateien von Websiten herunterzuladen [27]. Die Erweiterung an den Zeitpunkt der Arbeit nur für eine Firefox-Version vor Quantum verfügbar. Bei den USENIX Konferenzen war das kein schwieriges Unterfangen. Man öffnete alle relevanten Seiten mit den PDFs in Tabs und markierte die Seiten für den DownThemAll-Manager mit „All Tabs“. Als nächster Schritt wählte man beim Manager, das er nur die gefundenen PDF-Texte herunterlädt. Die Beschaffung der PDFs der SIGCOMM und der ANCS über den Bibliotheksproxy der Universität war begrenzt automatisiert möglich. Ein pro Nutzer festgelegtes, stündliches Downloadkontingent variiert je nach Verlag. Präventiv sperren Verlage den „e-Access“ für paar Stunden.

Nach der Beschaffung der Dokumente ist die Konvertierung der PDFs in TXT-Dateien wichtig um die weitere Bearbeitung zu gewährleisten. Da bietet PDFBox, eine open-source Java-Bibliothek, die notwendigen Methoden dazu bereit [28]. Unter der Angabe des Ordners konvertiert mittels for-each-Schleife jede PDF in eine .txt-Datei [29].

```
import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;
import org.apache.pdfbox.cos.COSDocument;
import org.apache.pdfbox.io.RandomAccessFile;
import org.apache.pdfbox.pdfparser.PDFParser;
import org.apache.pdfbox.pdmodel.PDDocument;
import org.apache.pdfbox.text.PDFTextStripper;

public class javaPDF2txt {

  public static void main(String[] args) {
    String pathOri = "Pfad des Ordners";
    File[] files = new File(pathOri).listFiles();
    for (File file : files) {
      if (file.isFile()) {
        System.out.println(file.getName());
        konvertieren(pathOri + "/" + file.getName(),
            file.getName());
      }
    }
  }

  public static void konvertieren(String path, String
      DateiName) {
    PDFTextStripper pdfStripper;
    PDDocument pdDoc;
    COSDocument cosDoc;
    File file = new File(path);
    try {
      RandomAccessFile randomAccessFile = new
          RandomAccessFile(file, "r");
      PDFParser parser = new PDFParser(randomAccessFile);
      parser.parse();
      cosDoc = parser.getDocument();
      pdfStripper = new PDFTextStripper();
      pdDoc = new PDDocument(cosDoc);
      String parsedText = pdfStripper.getText(pdDoc);
      try (PrintWriter out = new PrintWriter(DateiName +
          ".txt")) {
        // an Dateiname wird ein .txt angehaengt
        out.println(parsedText);
        pdDoc.close();
```


doi: 10.2313/NET-2018-11-1_07

```
      }
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

Da die Texte von Konferenzen stammen, die anderen Thematiken ansprechen und besprechen in den Veröffentlichungen sind viele Schriften nicht von Relevanz. Um dies herauszufinden, welche Dokumente von Bedeutung sind, nutzt der StringTokenizer der Java-Bibliothek. Diese Klasse bricht einen String in Tokens auseinander, dabei unterscheidet er nicht zum Beispiel zwischen Variablen und Zahlen. Dazu speichert man die Textdatei in einen String mithilfe eines Streams [31] und der speichereffizienten „Files.lines“ [32]. Mittels der Methoden „nextToken()“ und „contains()“ testet man, ob der Text das Keyword enthält und somit eine Relevanz zeigt.

```
public static void leser(String path) {
  String contents = "";
    try {
      contents = Files.lines(Paths.get(path)).
      collect(Collectors.joining());
      // Txt in einen String rein
    } catch (IOException e) {
      e.printStackTrace();
    }
  StringTokenizer st = new StringTokenizer(contents);
    while (st.hasMoreTokens()) {
      if (st.nextToken().contains("keyword")) {
        relevant = true;
      }
    }
}
```

Bei knapp 1605 Facharbeiten stellt sich dann heraus, dass in 108 Dokumente mindestens einer der vorher gennanten Frameworks erwähnt. Somit sind circa 7% für diese Ausarbeitung von Bedeutung. Damit der Überblick noch vorhanden bleibt, werden alle relevanten Texte in einen separaten Ordner kopiert, mittels der Methode „FileUtils.copyFile(source, destination)“.

7. ERGEBNISSE

Durch das gezielte Lesen von den Sätzen mit den Keywords war eine manuelle Zuordnung möglich, wie sich die Autoren mit den Frameworks auseinander gesetzt haben. Dabei wird unterschieden, ob der Verfasser die Framework erwähnt hat, oder sie implementiert hat und explizit im Text genannt wird. An diesem Punkt wird nochmal unterteilt, ob bei der Umsetzung auch das Backend miteinbezogen wird. Die Abbildung 3 zeigt die Verteilung der Publikationen, in wie vielen und welche Framework dabei genutzt wurde. Dabei stellt sich heraus, dass DPDK, netmap und PF_RING heraussstehen. Seit 2012, nach Veröffentlichung der Arbeit von Rizzo über netmap und im Jahre 2013, dass DPDK von einer opensource Gemeinschaft weiterentwickelt wird, gewannen beide Frameworks zugleich Anerkennung und Beliebtheit. Zudem fällt auf, dass netmap sehr oft genannt wird. Viele Artikel haben aus den Jahren 2015 und 2016 sich netmap als Inspiration genommen und lediglich nur in den Referenzen angeführt. Es existieren wenige Texte in den Jahren nach 2012,

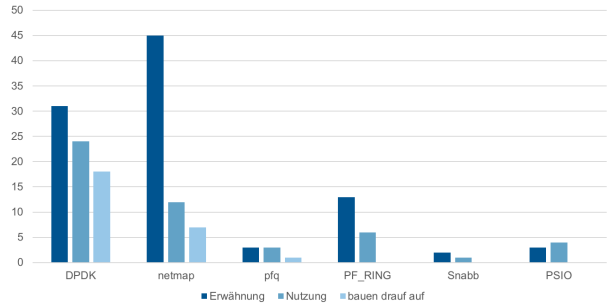

Abbildung 3: Auswertung 2010-2017

deren Inhalt nur eine Framework abhandelt. Wenn Beispiele von Paketverbeitungs-Frameworks gesucht werden, werden meist in den Publikationen DPDK und netmap genannt.

Nicht nur dieses Ergebnis ist interessant, sondern auch die Verteilung nach den Konferenzen nach den Frameworks, wie in Abbildung 4 zu sehen.

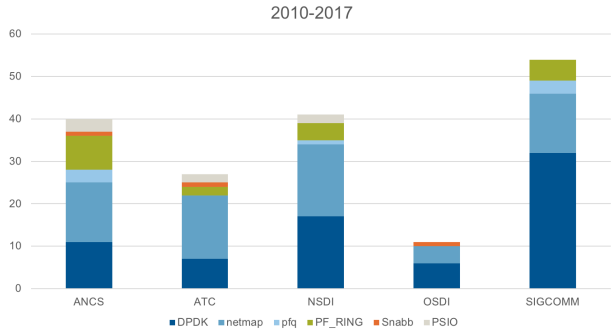

Abbildung 4: Auswertung nach Konferenz

Die SIGCOMM, die eine geringe Annahmequote bei Publikationen hat, neigen eher zu Intel DPDK. Die anderen zeigen entweder ein gleichwertiges Interesse zu DPDK und netmap, oder bevorzugen eher netmap in den Arbeiten. Zudem zeichnet sich heraus, dass die CPU-lastigen Frameworks überwiegen.

8. ZUSAMMENFASSUNG

Die Frameworks DPDK und netmap erweisen sich als gleichwertig populär heraus. Aber gerade in den letzten beiden Jahren schwenkt das Interesse für Implementationen von netmap mehr zu DPDK hin. Aber die Entwicklung Packet Processing Frameworks mit Hilfe von GPUs sollte man nicht ignorieren, weil sie neue Perspektiven geben in Themen wie Bandbreite und vermutlich auch später Performance. Folglich kann sich der Trend nochmal ändern und die zukünftigen Konferenzen dies zeigen.

9. LITERATUR

[1] *Globale Datenmenge wächst bis 2025 auf 163 Zettabyte | t3n*, `https://t3n.de/news/globale-datenmenge-2025-811914`, zuletzt besucht am 1. April 2018

[2] Dominik Schöffmann: *Comparing PFQ: A High-Speed Packet IO Framework*,

doi: 10.2313/NET-2018-11-1_07

https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2016-07-1/NET-2016-07-1_09.pdf, zuletzt besucht am 1. April 2018

[3] *DPDK*, http://dpdk.org/, zuletzt besucht am 1. April 2018

[4] *netmap*, http://info.iet.unipi.it/~luigi/netmap/, zuletzt besucht am 1. April 2018

[5] *05_SW_Architekturen*, https://wwwbroy.in.tum.de/lehre/vorlesungen/mbe/SS07/vorlfolien/05_SW_Architekturen.pdf, zuletzt besucht am 1. April 2018

[6] *Kernal Space Definiton*, http://www.linfo.org/kernel_space.html, zuletzt besucht am 1. April 2018

[7] *User Space Definition*, http://www.linfo.org/user_space.html, zuletzt besucht am 1. April 2018

[8] *Anatomy of the Linux kernel*, https://www.ibm.com/developerworks/linux/library/l-linux-kernel/, zuletzt besucht am 1. April 2018

[9] *What is Batch Processing? - Definition from Techopedia*, https://www.techopedia.com/definition/5417/batch-processing, zuletzt besucht am 1. Mai 2018

[10] *Diving into Snabb*, https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2016-09-1.pdf, zuletzt besucht am 1. April 2018, pages 31-38

[11] *snabb*, https://github.com/snabbco/snabb, zuletzt besucht am 1. April 2018

[12] *A Survey of Trends in Fast Packet Processing*, https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2014-08-1.pdf, zuletzt besucht am 1. April 2018, pages 41-48

[13] *Using GPUs for Packet-processing*, https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2014-08-1.pdf, zuletzt besucht am 1. April 2018, pages 33-40

[14] Rizzo, L.: *Netmap: a novel framework for fast packet I/O*, Proceedings of the 2012 USENIX conference on Annual Technical Conference, USENIX ATC'12, Berkeley, CA, 2012

[15] *PacketShader - GPU-accelerated Software Router*, https://shader.kaist.edu/packetshader/, zuletzt besucht am 2. April 2018

[16] *PFQ I/O*, https://www.pfq.io/, zuletzt besucht am 1. April 2018

[17] *PF_RING*, https://www.ntop.org/products/packet-capture/pf_ring/, zuletzt besucht am 2. April 2018

[18] *ACM History*, https://www.acm.org/about-acm/acm-history, zuletzt besucht am 31. März 2018

[19] *ACM Resources*, https://cacm.acm.org/acm-resources, zuletzt besucht am 31. März 2018

[20] *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, https://dl-acm-org.eaccess.ub.tum.de/citation.cfm?id=3098822, zuletzt besucht am 31. März 2018

[21] *IEEE - IEEE at a Glance*, https://www.ieee.org/about/today/at_a_glance.html, zuletzt besucht am 31. März 2018

[22] *ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, https://sites.google.com/a/ancsconf.org/www/, zuletzt besucht am 31. März 2018

[23] *About USENIX | USENIX*, https://www.usenix.org/about, zuletzt besucht am 31. März 2018

[24] *USENIX ATC '18 | USENIX*, https://www.usenix.org/conference/atc18#about, zuletzt besucht am 1. April 2018

[25] *NSDI '18 | USENIX*, https://www.usenix.org/conference/nsdi18, zuletzt besucht am 1. April 2018

[26] *OSDI '18 | USENIX*, https://www.usenix.org/conference/osdi18, zuletzt besucht am 1. April 2018

[27] *DownThemAll (or just dTa) is a powerful yet easy-to-use Mozilla Firefox extension that adds new advanced download capabilities to your browser*, https://www.downthemall.net, zuletzt besucht am 31. März 2018

[28] *Apache PDFBox | A Java PDF Library*, https://pdfbox.apache.org/, zuletzt besucht am 1. April 2018

[29] *java - How to extract text from a PDF file with Apache PDFBox - Stack Overflow*, https://stackoverflow.com/questions/23813727/how-to-extract-text-from-a-pdf-file-with-apache-pdfbox, zuletzt besucht am 1. April 2018

[30] *StringTokenizer (Java Platform SE 8)*, https://docs.oracle.com/javase/8/docs/api/index.html?java/util/StringTokenizer.html, zuletzt besucht am 1. April 2018

[31] *Java Streams - Collectors joining() example*, http://www.java2s.com/Tutorials/Java/java.util.stream/Collectors/Collectors.joining_.htm, zuletzt besucht am 1. April 2018

[32] *Java 8 API by Example: Strings, Numbers, Math and Files - Benjamin Winterberg*, http://winterbe.com/posts/2015/03/25/java8-examples-string-number-math-files, zuletzt besucht am 1. April 2018