Peer-to-Peer-Netzwerke

Was sind Peer-to-Peer-Netzwerke?
+ Umsetzung in einem Chat-Client mit Python

Florian Weißmeier-Dörste
Humboldtgymnasium Solingen Q1
Grundkurs Informatik G1
Fachlehrer: B. Pohler

Schuljahr 2020/21
10.02.2021

Inhaltsverzeichnis

1 Einführung

1.1 Thema

Im Folgenden möchte ich den Hintergrund meiner Themenentscheidung mit Ihnen teilen. Ich habe lange über mögliche Themen nachgedacht, mich letzendlich aber für P2P-Netwerke entschieden, da ich mich selbst sehr für dieses Thema interessiere. Außerdem wird diese Facharbeit dem Leser einen Gegensatz zu den üblichen Client-Server-basierten Netzwerkstrukturen bieten. In Absprache mit Herrn Pohler habe ich mich dazu entschieden, zur Veranschaulichung dieses Themas, einen Chat-Client in der Programmiersprache Python umzusetzen.
Dieses Thema ist im heutigen Zeitalter relevant, da durch das Ansteigen der Digitalisierung, in allen Teilen der Welt, immer mehr Computer miteinander verbunden werden. Es ist also wichtig zu wissen, welche Netzwerkstruktur für eine bestimmte Anwendung die beste ist und welche Vorteile sie hat.

1.2 Ziel

Das Ziel dieser Facharbeit ist es, eine alternative Netzwerkstruktur im Gegensatz zu Client-Server-Netzwerken darzustellen und zu erläutern. Der Leser soll nach dem Lesen der Facharbeit eine genaue Vorstellung davon haben, wie Peer-to-Peer-Netzwerke aufgebaut sind und welche Vorteile beziehungsweise Nachteile sie gegenüber anderen Strukturen haben. Diese Arbeit wird verschiedene Anwendungsfelder grundlegend erläutern, um dem Leser Beispiele zu zeigen, aber nicht tiefer analysieren. Außerdem wird in dieser Arbeit die Geschichte und die mögliche Zukunft von Peer-to-Peer-Netzwerken dargestellt.

1.3 Inhalt

Der Inhalt dieser Facharbeit besteht aus verschiedenen Kapiteln, um den Text in Unterthemen zu gliedern. Ich werde zu Beginn den Begriff "Peer-to-Peer Netzwerk" definieren und erklären. Darauf folgend werde ich die Geschichte von Peer-to-Peer-Netzwerken darstellen und die ersten Netzwerke beschreiben. Als viertes Unterthema werde ich Beispiele für Anwendungsfelder geben und diese erklären. Danach werde ich den von mir programmierten Chat-Client vorstellen und anhand diesem die Idee von Peer-to-Peer-Netzwerken aufzeigen. Zuletzt werde ich die Ergebnisse meiner Facharbeit zusammenfassen und diese beurteilen.

2 Definition

2.1 Grundlagen

Peer-to-Peer-Netzwerke, kurz P2P, sind Netzwerke aus Computern, sogenannte Peers, bei denen es keine zentralen Server gibt. Im Gegensatz zu Client-Server-basierten Strukturen sind Peer-to-Peer-Netzwerke also dezentral. Jeder Rechner im Netzwerk ist demnach „gleichberechtigt“. [1] Es gibt verschiedene Strukturen von Peer-to-Peer-Netztwerken. Je nach Struktur sind Netzwerke dezentraler oder organisierter. Bei großen Netzwerken zum Beispiel gibt es oft verschiedene Gruppen, in die Computer eingeteilt werden. So kann das Netzwerk besser organisiert werden. Außerdem wird bei großen Netzwerken, im Gegensatz zu kleinen, oft auch gespeichert, auf welchem Peer bestimmte Daten zu finden sind. Das ist teilweise auch notwendig, da sonst Anfragen an jeden verbundenen Peer geschickt werden müssten. Demnach steigt die Belastung beim Suchen nach Daten mit der Größe des Netzwerks. Bei Netzwerken die den Ort von Daten speichern und groß genug sind, um Computer in Gruppen zu unterteilen, ist oft eine Gruppe für das Finden von Daten zuständig. Bei Netzwerken wird außerdem auch nach dem Grad der Dezentralität unterschieden. Bei einem komplett dezentralen Netzwerk muss jeder verbundene Computer über alle Daten im Netzwerk Bescheid wissen. Bei einem teilweise dezentralen Netzwerk gibt es oft einen Computer, der die Rolle eines koordinierenden Servers einnimt. Dieser kann dann zum Beispiel die verbundenen Computer in Gruppen einteilen, Aufgaben verteilen oder Computer verwalten. Wenn man diese Struktur der weniger dezentralen Netzwerke verfolgt, kommt man früher oder später auch zu Super-Peer-Netzwerken. Dort werden sehr leistungsstarke Computer zu einem sogenannten Super-Peer. Dieser verwaltet das Netzwerk und kümmert sich um das Routing von Daten, was bei sehr großen Netzwerken hilfreich sein kann. Ein Beispiel für solch ein Netzwerk war Skype, bevor es 2016 seine Struktur größtenteils geändert hat.

2.2 Vorteile und Nachteile

Peer-to-Peer-Architekturen haben, je nach Anwendung, Vor- beziehungsweise Nachteile gegenüber anderen Infrastrukturen. Ein Vorteil ist beispielsweise die „hervorragende Skalierbarkeit“[2] von Peer-to-Peer-Netzwerken. Die verfügbaren Ressourcen eines Netzwerks können einfach mit mehr teilnehmenden

[1] [3] Z. 3
[2] [2]S. 3

Computern erweitert werden. Außerdem ist ein Peer-to-Peer-Netzwerk kaum von Ausfällen einzelner Computer betroffen. Diese Netzwerke sind also ziemlich ausfallsicher. Ein weiterer Vorteil ist, dass zu übertragende Daten direkt vom Sender zum Empfänger übertragen werden. In der Mitte ist kein Server, der die Daten weiterleitet und vielleicht sogar speichert. Im Hinblick auf die aktuellen Sorgen in Sachen WhatsApp und Ende-zu-Ende-Verschlüsselung sind Peer-to-Peer-Netzwerke eine sichere Alternative.

Auch wenn die Anzahl der Vorteile überwiegt, hat diese Netzwerkarchitektur auch Nachteile. Unter anderem muss jeder Computer im Netzwerk jederzeit über die Struktur Bescheid wissen. Wenn beispielsweise ein Computer aus dem Netzwerk entfernt wird, dann muss diese Information an alle anderen Computer im Netzwerk weitergegeben werden. Anderenfalls könnte ein Computer versuchen, mit einem anderen, nicht mehr existierenden, Computer zu kommunizieren. Um erneut auf Instant Messengers zurück zu kommen, muss man sagen, dass bei einem solchen Messenger, bestehend aus einem Peer-to-Peer-Netzwerk einige Funktionen eines Client-Server-Systems wegfallen. Wenn eine Person einer anderen auf WhatsApp eine Nachricht schreibt, dann wird diese erst einmal an einen WhatsApp-Server gesendet. Dort wird sie zwischengespeichert und an den Empfänger weitergeleitet, sobald dieser mit dem Internet verbunden ist. Sollte sein Endgerät gerade nicht verfügbar sein, dann wartet der Server, bis sich das Gerät bei ihm meldet und neue Nachrichten empfangen kann. Bei Peer-to-Peer-Systemen gibt es keinen zentralen Server, der die Nachrichten speichern und zustellen kann. Also müsste jeder, der diesen Messenger benutzt, jederzeit online sein, damit er alle Nachrichten erhalten kann.

2.3 Kritik

Peer-to-Peer-Netzwerke werden oft zum Teilen von Dateien genutzt. Allerdings werden nicht immer nur legale und harmlose Fotos oder Videos geteilt, sodern auch Raubkopien von Filmen oder Musik. Teilweise werden in solchen Netzwerken auch illegale pornografische Inhalte geteilt. Dadurch, dass in der Mitte kein zentraler Server steht, können diese Inhalte schwer überprüft werden. Viele dieser Netzwerke werden deshalb von den Behörden geschlossen, aber das sind weit nicht alle.

3 Geschichte und Zukunft

3.1 Geschichte

Das erste Peer-to-Peer-Netzwerk hat seinen Ursprung bei Shawn „Napster“ Fanning, der im Juni 1999 die erste Version seines gleichnamigen Netzwerks „Napster“ veröffentlichte. Der Gedanke hinter dieser Erfindung war „die Bereitstellung eines File-Sharing-Systems“[3], mit dem Nutzer Dateien direkt von Rechnern herunterladen konnten, welche ebenfalls in diesem System angemeldet sind. Durch einen Freund von Fanning wurde aus Napster ein Dienst zum Verbeiten von Musikdateien. Einige Musikverlage erhoben Anklagen gegen Fanning, da durch das illegale Teilen von Musik Urheberrechte missachtet wurde.

3.2 Zukunft

Es gibt viele Spekulationen über die Zukunft von Peer-to-Peer, aber eine ist teilweise bereits Realität. Cryptowährungen, die auf dem Blockchain-Prinzip beruhen, wie zum Beispiel Bitcoin oder Ethereum, haben vor ein paar Jahren das Interesse vieler Menschen geweckt. Bei solchen Blockchains wird auf das Distributed-Ledger-Prinzip zurückgegriffen, also ein Peer-to-Peer-Netzwerk, bei dem alle Transaktionen in einem dezentralen Verlauf gespeichert sind. Daduch ist es sehr schwer, wenn nicht sogar unmöglich, Einträge zu manipulieren oder zu ändern. Außerdem werden neue Transaktionen erst in die Blockchain aufgenommen, wenn midestens die Hälfte der teilnehmenden Computer diese Transaktion als legitim bewertet. Es stellt sich die Frage, ob diese Cryptowährungen die Zukunft des digitalen Bezahlens werden.

Experten gehen außerdem auch davon aus, dass in Zukunft große Peer-to-Peer-Netzwerke Supercomputer ersetzen könnten. Die Anfänge solcher Netzwerke entwickeln sich bereits auf Grund der aktuellen Pandemie. Projekte wie Folding@Home fordern Nutzer dazu auf, ihre ungenutzten Ressourcen zu Verfügung zu stellen, um, unter anderem, das SARS-CoV-2-Virus zu untersuchen. In Zukunft könnte dies der Menschheit helfen, sich schneller zu entwickeln und Viren schneller zu bekämpfen.

[3] [4] S. 4

4 Anwendungsfelder

4.1 Arbeitsgruppen

Das Prinzip des Peer-to-Peer-Netzwerks ist für kleine Arbeitsgruppen gut geeignet. Jeder kann seine Dateien oder angeschlossene Drucker mit seinen Arbeitskollegen teilen. Diese Freigaben werden nicht von einem zentralen Server verwaltet, sondern von jedem Nutzer einzeln. Protokolle für diese Freigaben wären zum Beispiel SMB bei Windows-Nutzern oder NFS bei Linux-Nutzern.

4.2 Datenaustausch

Peer-to-Peer eignet sich gut, um Dateien zu teilen und zu kopieren. Das kann im kleinen Stil, wie im Heimnetzwerk, geschehen, aber auch im großen, wie bei den vielen globalen Tauschbörsen.

4.3 Distributed Computing

Distributed Computing ist ein weiteres Anwendungsfeld von Peer-to-Peer. So können einzelne Computer ihre Rechenkapazitäten zusammenlegen, um größere Rechenoperationen schneller auszuführen. Jeder Computer erhält einzelne Teilaufgaben, welche zum Berechnen der gesamten Rechenoperation benötigt werden. Wenn alle Computer ihre Teilaufgaben berechnet haben, werden diese zu einem Ergebnis zusammengeführt. Dieses Anwendungsfeld wird hauptsächlich von wissenschaftlichen Institutionen benutzt.

5 Umsetzung als Chat-Client

5.1 Einführung

Ich habe mich dazu entschieden meinen, Peer-to-Peer Chat-Client mit Hilfe von Python umzusetzen. Python hat für solche Zwecke viele hilfreiche Bibliotheken und die Syntax ist einfach zu verstehen. Die folgenden, nicht vorinstallierten Bibliotheken habe ich benutzt, um den Chat umzusetzen:

- Socket
- Threading
- Curses
- Npyscreen

- Pyperclip

Ich habe zum Entwickeln des Chats die Versionsverwaltungssoftware Git benutzt und alle Versionen auf GitHub hochgeladen.

Das Programm benutzt zwei Sockets. Einen zum Empfangen (Server) und einen zum Senden (Client).

5.2 Dateifunktionen

Im Folgenden werde ich die Funktion aller Dateien erklären.

Angefangen mit run.py. Generell gesehen hat run.py erstmal keine, für den Chat relevanten Funktionen. Mit Hilfe von run.py kann der Benutzer alle nötigen Bibliotheken installieren, falls er den Packetmanager pip installiert hat.

Sollten alle Module installiert sein, wird chat.py ausgeführt. Chat.py ist das Herz des Chats. Es importiert meine Client- und Server-Module, auf die ich gleich weiter eingehen werde, und erstellt ein Client- und ein Server-Objekt. Beide benutzen Sockets ,um sich mit einem Peer zu verbinden. Chat.py importiert außerdem auch ein Command-line-interface (CLI) aus der Datei form.py. Das Interface ist simpel gehalten und hat ein Texteingabefeld und ein Textanzeigefeld. Das Interagieren mit der Software basiert auf Befehlen. Jeder Befehl wird mit einem »/« eingeleitet und kann mehrere Argumente haben. Ein Beispiel für einen solchen Befehl wäre »/connect [host] [port]« . Wie man sehen kann, nimmt der Befehl zwei Argumente entgegen. Eimal Host, mit dem ein Zielcomputer angegeben wird und einmal Port, mit dem der Port des Zielcomputers angegeben wird. Standardmäßig läuft der Chat auf port 3333, kann aber mit »/port [port]« geändert werden. Weitere Befehle sind auf GitHub nachzulesen.

Wie bereits erwähnt, enthält form.py das CLI. Ich benutzte eine Bibliothek namens npyscreen, welche es erlaubt, eigene Benutzerschnittstellen innerhalb eines Terminals zu bauen.

In der client.py Datei wird der Client des Chats definiert. Client.py erstellt einen neuen Socket und wartet auf den »/connect« Befehl. Wenn der Nutzer sich mit Hilfe des Befehls mit einem anderen Nutzer verbindet, kann Chat.py Nachrichten senden und empfangen.

Ähnlich wie client.py erstellt server.py erstmal einen neuen Socket. Danach wartet dieser Socket auf Verbindungen von einem anderen Nutzer. Wenn sich ein Nutzer zum Server verbindet, prüft server.py, ob der Client-Socket bereits verunden ist. Wenn nicht, kann sich der Nutzer mit »/connectback« zu dem Nutzer verbinden, der sich zu seinem Server verbunden hat. Außerdem wartet der Server auf neue Nachrichten von verbundenen Clients.

Verbundene Clients und Server können miteinander kommunizieren, indem sie spezielle Befehle benutzen. Diese werden dann an den verbundenen Server gesendet und von diesem als Serverbefehle interpretiert. So kann zum Beispiel die Software verbundenen Systemen mitteilen, dass der Nutzer seinen Spitznamen mit »/nick [name]« geändert hat.

Sowohl Client als auch Server haben die Oberklasse Thread aus dem Modul Threading. Python ist so programmiert, dass immer nur ein Befehl auf einmal ausgeführt werden kann (Singlethread). Das Programm könnte also nicht gleichzeitig empfangen und senden. Das Threading Modul erlaubt es Python, mehrere einzelne Threads auszuführen (Multithreading). So kann der Chat gleichzeitig senden, empfangen und Befehle entgegennehmen.

Dateien im /lang/-Ordner sind Sprachdateien. Sie beinhalten Übersetzungen aller Systemausgaben in einem JSON-Objekt und können im Chat mit »/lang [sprache]« geladen werden. Nutzer können eigene Sprachen hinzufügen und diese im Chat verwenden. Die Namen von Sprachdateien richten sich nach dem ISO-639-1 Standard um es dem Nutzer zu erleichtern, die Namen in Erinnerung zu behalten.

In der settings.json-Datei werden die vom Nutzer gewählten Einstellungen gespeichert. Unter anderem die zuletzt eingestellte Sprache.

Genauere Beschreibungen einzelner Methoden und Objekte sind innerhalb des Codes in Form von Kommentaren zu finden.

Eine Installationsanleitung und eine Liste mit genauen Beschreibungen der Befehle sind auf GitHub zu finden.

5.3 Nutzung

Um den Chat-Client zu starten, sollte run.py ausgeführt werden, um zu überprüfen, ob alle benötigten Module installiert sind. Wenn dies nicht der Fall ist, wird run.py versuchen, die fehlenden Module zu installieren. Alternativ kann man aber auch direkt chat.py ausführen, um diese Überprüfung zu übersprin-

gen. Wenn das Programm gestartet wurde, sollte man das CLI sehen.

Zur Einstellung des eigenen Serverports kann man »/port [port]«verwenden. Der Server wird daraufhin auf dem angegebenen Port neu gestartet.

Bevor man sich mit einem anderen Peer verbinden kann, muss man seinen Spitznamen mit »/nick [name]« festlegen. Danach steht dem Chatten nichts mehr im Weg. Mit »/connect [host] [port]« kann man sich jetzt mit einem anderen Peer verbinden. Wichtig zu wissen ist, dass auf beiden Geräten der festgelegte Port in der Firewall zugelassen und im gegebenenfalls Router weitergeleitet werden muss. Sobald sich einer der beiden Chatpartner verbunden hat, kann der andere das Eingeben des »/connect« Befehls überspringen, indem er »/connectback« benutzt. Dieser Befehl verbindet den Client direkt mit dem Peer, der bereits mit dem Server verbunden ist. Wenn beide Partner mit einander verbunden sind, können Nachrichten ausgetauscht werden. Einfach eine Nachricht in das Eingabefeld eingeben und Enter drücken. Sollte die Anzahl von Nachrichten die maximale Höhe des Textanzeigefelds überschreiten, wird dieses geleert. Alle Nachrichten der Sitzung werden allerdings auch in einem Chatlog gespeichert, der mit »/log« als Datei gespeichert werden kann. Sollte man eine bereits gesendete Nachricht erneut senden wollen, kann man mit den Pfeiltasten durch den Verlauf der gesendeten Nachrichten navigieren.

Zum Beenden eines Chats kann man den »/disconnect«-Befehl benutzen. Er trennt die Verbindungen der Sockets und erstellt diese neu. Alternativ kann man »/quit« schreiben, um die App komplett zu beenden.

Für Entwickler, die das Programm debuggen wollen, steht der »/eval«-Befehl zu Verfügung. Mit ihm kann man Python-Code innerhalb der App ausführen. Das kann nützlich sein, um den aktuellen Wert von Variablen herauszufinden, oder um Methoden per Hand zu starten. Ein Beispiel dafür wäre »/eval print(self.port)«, um den aktuellen Port auszugeben.

6 Fazit

6.1 Zusammenfassung

Zusammengefasst sind Peer-to-Peer-Netzwerke solche, bei denen verbundene Computer (Peers) gleichberechtigt sind. Es gibt keinen zentralen Server, sondern, und das auch nur bei großen Netzwerken, nur Server zum Managment des Systems. Peer-to-Peer-Netzwerke sind sicherer und weniger ausfallgefährdet als Client-Server-Netzwerke. Allerdings müssen alle Peers stets über den

aktuellen Stand des Systems Bescheid wissen. Benutzt werden solche Systeme zum Beispiel für Datenaustausch, kleinere Firmen mit Arbeitsgruppen und zum Ressourcenteilen. Die Kritik am Peer-to-Peer-System ist, dass viele Menschen illegal Dateien über Peer-to-Peer-Netzwerke teilen, darunter auch Filme oder pornografische Inhalte. Die Behörden arbeiten seit Jahren daran, diese Tauschbörsen zu schließen. In Zukunft können Peer-to-Peer-Netzwerke als Ersatz für Supercomputer genutzt werden oder um Transaktionen sicher zu genehmigen.

6.2 Reflexion

Ich habe mit dieser Arbeit die Eigenschaften von Peer-to-Peer-Netzwerken sowie die Geschichte und die mögliche Zukunft dieser dargestellt. Die Anwendungsfelder dieser Netzwerke wurden erklärt und teilweise auf die aktuelle Situation bezogen. Die Arbeit hat die Vorteile, aber auch die Nachteile von Peer-to-Peer-Netzwerken gegenüber anderen Netzwerken aufgezeigt und erläutert. Insgesamt wurden die nötigen Fragen zum Verständnis von Peer-to-Peer-Netzwerken beantwortet.

6.3 Ausblick

Um zu verstehen, wann welche Netzwerkstrukturen gebraucht werden und welche Vorteile der Gebrauch der jeweiligen Struktur gegenüber allen anderen hat, sollte man sich genauer mit anderen Netzwerkstrukturen auseinandersetzen. Das Wissen über mehrere Strukturen zu haben, ermöglicht es für, eine bestimmte Anwendung eine optimale Netzwerkstruktur zu finden.

Literatur

[1] IONOS by 1&1. *Was ist eine Blockchain? Definition & Funktionsweise*. 27. Juli 2018. URL: https://www.ionos.de/digitalguide/online-marketing/verkaufen-im-internet/blockchain/ (besucht am 23. 01. 2021).

[2] Stefan Dipl.-Ing. (FH) Luber. *Was ist Peer-to-Peer (P2P)?* 1. Aug. 2018. URL: https://www.ip-insider.de/was-ist-peer-to-peer-p2p-a-654713/ (besucht am 20. 01. 2021).

[3] Klaus Lipinski. *Peer-to-Peer-Netz*. 24. Apr. 2020. URL: https://www.itwissen.info/Peer-to-Peer-Netz-peer-to-peer-network-P2P.html (besucht am 13. 01. 2021).

[4] Peter Mahlmann und Christian Schindelhauer. *Peer-to-Peer-Netzwerke: Algorithmen und Methoden*. 1. Aufl. Leverkusen: Springer, 2007. ISBN: 978-3-540-33991-5.

[5] ryte.com. *Peer-to-Peer*. 17. Jan. 2021. URL: https://de.ryte.com/wiki/Peer-to-Peer (besucht am 17. 01. 2021).

[6] Hans Röck. *Peer-to-Peer-Netzwerk*. 18. Feb. 2013. URL: https://www.enzyklopaedie-der-wirtschaftsinformatik.de/wi-enzyklopaedie/lexikon/technologien-methoden/Rechnernetz/Datenkommunikation/Peer-to-Peer-Netzwerk (besucht am 13. 01. 2021).

[7] Xovi.de. *Was bedeutet Peer-to-Peer?* 18. Feb. 2013. URL: https://www.xovi.de/was-bedeutet-peer-to-peer/ (besucht am 13. 01. 2021).

Selbstständigkeitserklärung

Ich versichere, dass ich die Facharbeit selbstständig verfasst, dass ich keine anderen Quellen und Hilfsmittel als die angegebenen benutzt und die Stellen der Facharbeit, die ich anderen Werken im Wortlaut oder dem Sinn nach entnommen habe, in jedem Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Solingen, den 10. Februar 2021

_____

Florian Weißmeier-Dörste

Arbeitsbericht

Datum	Uhrzeit	Aktion
05.10.2020	-	1. Facharbeitsworkshop
12.12.2020	-	2. Facharbeitsworkshop
06.01.2021	15:30 bis 16:30	Quellenfindung
17.01.2021	11:00 bis 14:30	Layout und Inhaltsstruktur
20.01.2021	12:00 bis 12:45	Weiterarbeiten an Kapitel 3, 4.3, 1.1
23.01.2021	12:00 bis 14:30	Weiterarbeiten an allen Kapiteln
31.01.2021	20:00 bis 20:45	Weiterarbeiten an Chat-Dokumentation
07.02.2021	11:30 bis 14:00	Weiterarbeiten an Fazit
07.02.2021	15:00 bis 15:30	Korrektur von Fehlern
10.02.2021	16:00 bis 16:30	Vollendung

Tabelle 1: Arbeitsbericht