Zentrum für sichere Informationstechnologie – Austria
Secure Information Technology Center – Austria

Seidlgasse 22 / 9, 1030 Wien
Tel.: (+43 1) 503 19 63–0
Fax: (+43 1) 503 19 63–66

Inffeldgasse 16a, 8010 Graz
Tel.: (+43 316) 873-5514
Fax: (+43 316) 873-5520

http://www.a-sit.at
E-Mail: office@a-sit.at

DVR: 1035461
ZVR: 948166612
UID: ATU60778947

VERTEILTER DATENSPEICHER IN HETEROGENEN UMGEBUNGEN

Version 1.0 vom 11.07.2018
Bernd Prünster – bernd.pruenster@a-sit.at

Abstract/Zusammenfassung: Geräteübergreifende, gemeinsame Datenspeicher sind vor allem in Form von cloudbasierten Diensten etabliert. Übliche Alternativen zu diesen oftmals serverzentrischen Architekturen ermöglichen typischerweise entweder die einheitliche Verteilung oder die gleichmäßige Replikation von Datensätzen innerhalb eines Geräteverbunds. Im Rahmen dieses Projekts wurden Lösungsansätze erarbeitet, welche die Speicherplatzheterogenität von Endgeräten in Rahmen verteilter Datenspeicher berücksichtigt. Das Ergebnis dieser Arbeit ist eine speicherplatzagnostische Weiterentwicklung des Kademlia-Protokolls, welches die Lücke zwischen homogen verteilten Speichermodellen wie z.B. klassische verteilte Hashtabellen (Distributed Hash Tables, DHTs) und reinen Cloud-speicherdiensten schließt. Insbesondere ist das erarbeitete Protokoll mit aktuellen (besonders im Peer-to-Peer-Kontext wichtigen) Sicherheitsmechanismen kompatibel. Darüber hinaus wurde auch der Einsatz von Cache-Replacement-Strategien im Kontext von verteilten Netzwerkspeichern beleuchtet.

Inhaltsverzeichnis

1. Einleitung

Viele aktuell populäre, vernetzte Anwendungen setzen einen geräteübergreifenden Speicher, bzw. eine gemeinsame, geräteübergreifende Datenbank voraus. Für sich betrachtet wird diese Anforderung üblicherweise in Form serverzentrischer Cloud-Lösungen, und dezentral durch verteilte Hashtabellen (Distributed Hash Tables, DHTs), oder vollständige Replikation gelöst. Ziel dieses Projekts ist die Konzeption eines dezentralen, verteilten Speichers, welcher die Speicherplatzheterogenität aktueller Endgeräte berücksichtigt. Weder Replikation noch klassische DHTs erfüllen diese Anforderung. Im ersten Fall kann lediglich selektiv repliziert werden, um auch Geräte mit begrenztem Speicherplatz zu bedienen, während DHTs üblicherweise eine vollkommen homogene Speicherplatzverteilung annehmen. Besonders letzteres spiegelt jedoch nicht die aktuellen Verhältnisse wie die Popularität unterschiedlichster Geräteklassen (Smartphones, Desktop-Computer, Tablets, Laptops, …) wider. Um dies auch in verteilten Speichermodellen abzubilden, muss die Lücke zwischen vollständiger, bzw. selektiver Replikation und probabilistischer Gleichverteilung von Daten im Rahmen von DHTs geschlossen werden: Daten müssen wie im Rahmen von DHTs (bestenfalls dedupliziert) auf Geräte verteilt werden; gleichzeitig ist es notwendig, dass von einer Anwendung aktuell benötigte Daten auf allen Geräten verfügbar sind. Aus Sicht der Anwendung, welche auf die verteilten Daten zugreift, ist es irrelevant, wie diese verwaltet werden. Die einzige Anforderung ist, dass Daten verfügbar sind, wenn sie benötig werden. Aus Nutzersicht ergeben sich allerdings sehr wohl weitere Anforderungen, wie beispielswiese zufriedenstellende Performance und niedrige Latenzen. Berücksichtigt man insbesondere auch Geräte wie Smartphones und Laptops, ist Effizienz ein wichtiges Kriterium, um den Stromverbrauch durch unnötige Rechenoperationen gering zu halten. Effizienz ist darüber hinaus auch bezüglich des verursachten Netzwerkverkehrs potentiell kritisch, da hohe Netzwerklast besonders im Mobilbereich zu schlechter Performance oder unerwünschtem Verbrauch von Datenvolumen führen kann.

Im Rahmen dieses Dokuments werden Konzepte vorgestellt, welche es ermöglichen, Daten innerhalb eines Netzwerks je nach verfügbarem Speicherplatz, derart auf Endgeräten zu verteilen, dass Geräte mit weniger Speicherplatz auch weniger Daten speichern müssen. Um dies zu erreichen, gibt es unterschiedliche Herangehensweisen. In jedem Fall ist für eine dezentrale Umsetzung jedoch ein dezentrales Routingverfahren notwendig, um die beteiligten Endgeräte vernetzen zu können. Hierfür haben sich Distributed Hash Tables über viele Jahre etabliert. Darüber hinaus bieten DHTs die Möglichkeit, Daten im Netzwerk zu speichern, wobei diese traditionell auf homogene Speicherplatzverteilung ausgelegt sind. Grundlegenden Techniken für verteiltes Routing können jedoch ungeachtet dessen verwendet werden.
Eine weitere Herangehensweise an das Problem der speicherplatzagnostischen Verteilung von Daten in einem Netzwerk mit heterogener Speicherplatzverteilung basiert auf Techniken welche primär im Rahmen der Speicherverwaltung von Mikroprozessoren eingesetzt werden und auf den Einsatz in einem Geräteverbund umgelegt und adaptiert werden können. Derartige Ansätze sind insbesondere nützlich, wenn darauf Wert gelegt wird, dass möglichst diejenigen Daten, die aktuell benötigt werden, auf allen Geräten verfügbar sind. Andere Daten, welche voraussichtlich nicht akut lokal verfügbar sein müssen, können hingegen weiterhin mittels traditioneller Distributed Hash Tables auf Geräte verteilt und bei Bedarf abgerufen werden.

Details zu den Hintergründen werden im nachfolgenden Abschnitt dargelegt. Die Abschnitte 2.1 und 2.2 gehen im Detail auf die unterschiedlichen Konzepte ein und Abschnitt 4 beschreibt die Erarbeitung eines konkreten Konzepts zur speicherplatzagnostischen Verteilung von Daten in verteilen Hashtabellen. Abschließend wird ein Ausblick auf eventuelle Weiterentwicklungen der vorgestellten Ansätze, sowie zusätzliche Anwendungsmöglichkeiten gegeben.

2. Hintergrund

Unabhängig davon, wie Daten innerhalb eines Netzwerks verteilt werden sollen, müssen Mechanismen vorhanden sein, welche eine direkte Vernetzung der involvierten Geräte ermöglichen. Hier haben sich insbesondere Distributed Hash Tables etabliert, weshalb deren Funktionsweise, dezentrale Routingverfahren und insbesondere deren Eigenschaften als verteilter Datenspeicher nachfolgend beschrieben werden. Anschließend beschreibt Abschnitt 2.2 Speicherverwaltungstechniken von Mikroprozessoren, welche ebenfalls auf verteilte Datenspeicher umgelegt werden können.

2.1. Routing und Datenspeicherung in Distributed Hash Tables

Distributed Hash Tables umfassen ein Klasse von Peer-to-Peer (P2P)-Netzwerken. Beispiele dafür sind *BitTorrent* [1] und *IPFS*[1]. Diese ermöglichen durch die Trennung von Ort und Identität die Verbindung von Geräten im Rahmen logischer Overlay-Netze auf Basis vorhandener heterogener Netzwerke. Die Adressierung von Netzwerkknoten erfolgt über logische, von der Netzwerkadresse entkoppelte Identifikatoren. Dadurch entsteht ein flacher Adressraum. Um eine Verbindung zu einem mittels Identifikator festgelegten Zielknoten aufzubauen, kommen spezielle verteilte Overlay-Routingprotokolle zu Einsatz, welche die zu einem Identifikator gehörige Netzwerkadresse auflösen. Besonders in vollständig dezentralen P2P-Netzwerken kann jedoch die Validierung von Herkunft und Authentizität von Daten (und Routinginformationen) zu einer Herausforderung werden [2] [3] [4] [5] [6] [7] [8]. Diesbezügliche Details und Lösungsansätze können gesammelt einem früheren A-SIT-Projekt zum Thema sichere Peer-to-Peer-Netze auf Basis von Selbstzertifizierung [9] entnommen werden. Zusätzlich geht Abschnitt 4.4 zusammenfassend auf entsprechende Sicherheitsaspekte ein, welche besonders im Zuge des entwickelten speicherplatzagnostischen Routingverfahrens relevant sind.

Allen (dezentralen) Peer-to-Peer-Netzen ist gemein, dass das Overlay-Netzwerk betreffende Routinginformationen von den Netzwerkteilnehmern selbst erzeugt und verbreitet werden. Jeder Knoten im Netzwerk verfügt über unabhängig vom Rest des Netzwerks verwaltete Routingtabellen, welche logische Identifikatoren auf die Netzwerkadresse des zugehörigen Knotens abbilden.
Routingtabellen werden wie im Rahmen traditioneller Routingprotokolle durch Routinganfragen, bzw. Antworten auf Routinganfragen befüllt. Ausgehend von einer (implementierungsabhängigen) Distanzfunktion, welche es ermöglicht, Distanzen zwischen logischen Identifikatoren zu berechnen, werden Routinganfragen wie folgt abgearbeitet: Um Informationen zu einem Zielknoten – die dem Identifikator des Zielknotens zugeordnete Netzwerkadresse – zu erhalten, wird eine Untermenge der bereits bekannten Knoten kontaktiert. Dabei handelt es sich aus Sicht des anfragenden Knotens um die dem Zielknoten am nächsten liegenden Knoten. Diese antworten ihrerseits mit Informationen zu ihnen bekannten Knoten, welche wiederum am Zielknoten liegen. Dieser Prozess wird iterativ so lange wiederholt, bis die Anfragen am Zielknoten konvergieren und dessen Netzwerkadresse bekannt ist.

Dasselbe Prinzip kann auch dafür verwendet werden, um Daten im Netzwerk zu verteilen und abzurufen: Verteilte Hashtabellen sind eine Form von *Content-Addressable Storage* – Daten werden über ihren Inhalt adressiert und nicht etwa an Hand von Metadaten. Beispiele hierfür sind *Kademlia* [10], *Chord* [11], oder *CAN* [12], aber auch aktuelle Entwicklungen, wie beispielsweise IPFS. Bei dieser Art von Netzwerken teilen sich Teilnehmer und Daten denselben Adressraum. Dies wird üblicherweise erreicht, indem Identifikatoren von Daten durch die Anwendung einer kryptografischen Hashfunktion auf ebendiese Daten errechnet werden. Wenn der resultierende Hash in derselben Domäne wie die Identifikatoren der Knoten liegt, ist es möglich „Routinganfragen“ auf Daten abzusetzen, sofern deren Hash bekannt ist. In diesem Fall konvergiert eine Anfrage an dem den Daten nächstgelegenen Knoten, welcher gleichzeitig für die Speicherung der Daten zuständig ist. Bei Gleichverteilung von Knoten- und Daten-Identifikatoren ergibt sich eine Gleichverteilung von Daten über alle Netzwerkteilnehmer. Insgesamt können somit sowohl Daten als auch Knoten durch effiziente ($\mathcal{O}(\log n)$) deterministische Verfahren lokalisiert werden. Manche Umsetzungen unterstützen darüber hinaus Replikation, sodass Daten (nach wie vor gleichverteilt) mehrfach abgespeichert werden, um die Ausfallsicherheit zu erhöhen.
Abgesehen von einigen speziellen Anwendungsfällen spiegelt derartige Datenverteilung jedoch kaum die Verteilung von Speicherkapazität in DHTs wider, da diese üblicherweise von Geräten unterschiedlichster Konfiguration aufgespannt werden. Indirekte Adressierung schafft hier teilweise Abhilfe: Anstatt Daten direkt zu speichern, werden lediglich Referenzen im Netzwerk gleichverteilt,

[1] https://ipfs.io/

welche über das zuvor beschriebene Verfahren abgerufen werden können. Eine Referenz verweist auf den Teilnehmer, welcher die gesuchten Daten tatsächlich speichert. Dadurch lässt sich steuern, wo welche Daten gespeichert werden. Da jedoch der Knoten, welcher Daten in das Netzwerk einbringt, entscheidet, wo diese gesichert werden, bzw. je nach Implementierung lediglich Referenzen auf sich selbst als Speicherort verteilen kann, ergibt sich dadurch keine direkte Möglichkeit, einen gemeinsamen Datensatz so auf alle Teilnehmer zu verteilen, dass eine potentielle (bzw. in Anbetracht des aktuell heterogenen Geräteumfelds, anzunehmende) Speicherplatz-heterogenität berücksichtigt wird; Daten also dort gespeichert werden, wo Speicherplatz vorhanden ist. Die bloße Berücksichtigung dieses Umstands bringt jedoch nur einen begrenzten Mehrwert mit sich: Werden Daten lediglich an Hand von Speicherplatzverfügbarkeit verteilt, ist nicht sichergestellt, dass Daten auch dort gespeichert werden, wo sie zu einem gegebenen Zeitpunkt benötigt werden. Hier können wohldefinierte Replikationsmechanismen Abhilfe schaffen.
Zu jedem gegebenen Zeitpunkt müssen idealerweise all jene Daten auf die Knoten repliziert werden, die diese Daten aktuell benötigen. Dieser Ansatz ist insbesondere für Arbeitsabläufe geeignet, welche auf allen beteiligten Geräten zeitlich verteilt ausgeführt werden. Wann welche Daten wo repliziert werden müssen, lässt sich auf Basis bestehender Algorithmen vorhersagen, welche üblicherweise als Teil der Speicherverwaltung von Mikroprozessoren zum Einsatz kommen: Mehrstufige Speicherhierarchien (insbesondere Hauptspeicher, und mehrere Cache-Ebenen) erreichen unter anderem erst durch den Einsatz von Prefetching und Cache-Replacement-Strategien hohe Performance. Dieses Modell lässt sich für zeitlich verteilte Prozesse auch auf verteile Netzwerkspeicher umlegen.

2.2. Cache-Replacement-Strategien im Kontext von verteiltem Netzwerkspeicher

Die Notwendigkeit von Cache-Replacement-Strategien im Rahmen der Speicherverwaltung von Mikroprozessoren ergibt sich aus der Speicherhierarchie bestehend aus mehreren Cache-Ebenen und Hauptspeicher. Ausgehend vom First-Level-Cache bringt jeder Schritt nach oben in dieser Hierarchie mehr Speicherplatz aber auch höhere Latenzen mit sich. Hohe Performance wird dann erreicht, wenn der vorhandene Speicher auf jeder Ebene optimal genutzt wird. Um dies zu bewerkstelligen, ist es einerseits notwendig, Daten vorab aus langsameren Ebenen zu laden (Prefetching), andererseits muss gleichzeitig entschieden werden, welche Daten in langsamere Speicherebenen ausgelagert werden, sobald eine Ebene voll belegt ist. Für beide Problemstellungen wurden eine Reihe unterschiedlicher Algorithmen entwickelt. Das darunterliegende Prinzip lässt sich in einem ersten Schritt unabhängig davon ebenfalls auf die Kombination von einer begrenzten Menge lokalem Speicherplatz und einem umfangreichen Netzwerkspeicher umlegen. Dadurch ergibt sich eine Speicherhierarchie bestehend aus zwei Ebenen, auf die dieselben Prinzipien anwendbar sind.
Daten werden bei Bedarf vom Netzwerk geladen und lokal gespiegelt. Ist der lokale Speicherplatz belegt, werden aktuell nicht benötigte Daten lokal entfernt und im Netzwerkspeicher vorgehalten. Dieses Szenario lässt sich im nächsten Schritt auf mehrere Geräte, die auf einen gemeinsamen Netzwerkspeicher zugreifen, erweitern. Will man erreichen, dass Aktivitäten, die auf einem Gerät gestartet werden auf anderen Geräten fortgesetzt werden können, ist es notwendig, dafür zu sorgen, dass auf einem Gerät vorgehaltene Daten ebenfalls auf allen anderen Geräten verfügbar sind. Berücksichtigt man die Speicherplatzheterogenität innerhalb aller Geräte, bedeutet das, dass die Menge lokal verfügbarer Daten unterschiedlich ist, in jedem Fall jedoch aktuell benötigte Daten im Sinne passender Cache-Replacement-Strategien bis zum (je nach Gerät unterschiedlichen) Speicherplatzlimit erhalten bleiben. Dadurch werden verteilte Arbeitsabläufe ermöglicht, die auf den gesamten Datenbestand zugreifen, ohne dass benutzerseitig festgelegt werden muss, wo welche Daten gespeichert werden sollen. Eine mögliche Strategie ist *Least-Recently Used* (LRU), welche diejenigen Daten auslagert, auf welche am längsten nicht mehr zugegriffen wurde. Ersetzt man in einem letzten Schritt den Netzwerkspeicher beispielsweise gegen eine DHT, welche von allen Geräten im Verbund aufgespannt wird, ergibt sich ein vollkommen dezentrales Szenario. Damit die Speicherplatzheterogenität zwischen den Geräten auch auf DHT-Ebene reflektiert wird, muss jedoch sichergestellt werden, dass der über die DHT verteilte Datensatz nicht auf alle Teilnehmer gleichverteilt wird. Konzepte mit diesem Ziel werden im nachfolgenden Abschnitt erläutert.

3. Bestehende speicherplatzagnostische DHT-Konzepte

Bereits Anfang der Zweitausenderjahre gab es Bemühungen, DHTs zu erweitern, um Ressourcenheterogenität im Netzwerk zu berücksichtigen [13] [14]. Im Wesentlichen gibt es zwei unterschiedliche Ansätze der Gleichverteilung von Last auf alle Netzwerkteilnehmer zu begegnen: Entweder man adaptiert die Distanzfunktion der DHT, sodass diese kapazitätsabhängig wird, oder man führt virtuelle Knoten ein. Letzteres bedeutet schlichtweg, dass jedes Gerät dem Netzwerk gegenüber nicht mehr als ein einziger Knoten auftritt, sondern mehrere Knoten betreibt – wie viele hängt von der Kapazität des Geräts ab.

In diesem Kontext wurde bewusst *Kapazität* behandelt, da insbesondere die Verwendung virtueller Knoten nicht nur zu einer Ungleichverteilung von Daten, sondern je nach Knotenanzahl auch zu mehr Last und Einfluss im Netzwerk führt. Beispielsweise kann (bzw. muss) ein Gerät durch das Betreiben von unverhältnismäßig vielen Knoten einen signifikanten Teil des gesamten Netzwerks kontrollieren und kann dieses dadurch unterwandern. Dies wird als *Sybil-Attacke* [15] bezeichnet und kann gesamte Netzwerke destabilisieren, da ein solcher Angriff als Basis für weitere Attacken dienen kann.
Unabhängig vom Sicherheitsaspekt bringen virtuelle Knoten schlicht zusätzliche Last mit sich. Können Routinganfragen ohne virtuelle Knoten noch in effizienten $\mathcal{O}(\log n)$ vielen Schritten abgearbeitet werden, vervielfacht sich dieser Aufwand abhängig von der durchschnittlichen Anzahl virtueller Knoten. Zwar gibt es Ansätze, diesen Overhead zu minimieren, indem virtuelle Knoten in Clustern zusammengefasst werden, allerdings setzt dies üblicherweise eine (in bestimmten Grenzen) freie Wahl von Identifikatoren voraus. Freie Identifikatorwahl schafft jedoch die Möglichkeit, *Eclipse-Attacken* [3] auszuführen. Das Ziel dieser Klasse von Attacken ist das Einkesseln, bzw. Ausgrenzen einzelner Netzwerkknoten (oder in der DHT gespeicherter Daten an Hand ihrer Identifikatoren). Erreicht wird dies eben durch die gezielte Erstellung von Identifikatoren, sodass Knoten derart in der Netzwerktopologie platziert werden, dass diese ausgewählte Zielknoten vollständig einkreisen. In Folge dessen ist immer zumindest ein bösartiger Knoten Teil eines jeden Routingpfads in Richtung Zielknoten. Ausgehend von dieser Position können Routinganfragen das Angriffsziel betreffend verfälscht, bzw. verworfen werden. Auf diese Art und Weise lassen sich Knoten vollständig vom restlichen Netzwerk abschneiden. Folglich sind Ansätze auf Basis virtueller Knoten mit Clustering nicht praktikabel, um die Effekte erhöhte Netzwerkgröße auf Grund virtualisierter Knoten zu minimieren.

Aus den zuvor genannten Gründen spricht vieles für die Adaptierung der in einer DHT eingesetzten Distanzfunktion, sodass Knoten mit mehr Speicherplatz öfter für Daten zuständig sind. Auch unabhängig von Sicherheitsbedenken wurde in der Literatur gegen virtuelle Knoten argumentiert [16]. Bestehende Forschungsarbeiten in diese Richtung liegen allerdings über zehn Jahre in der Vergangenheit [13] [14] und sind daher nur bedingt auf die mittlerweile geänderten Gegebenheiten anwendbar. Insbesondere hat sich in der Zwischenzeit Kademlia als Modell für verteilte Hashtabellen durchgesetzt, während frühe Konzepte für heterogene Ressourcenverteilung auf CAN oder Chord aufbauen. Nachfolgend wird ein auf Kademlia basierendes Modell für eine speicherplatzagnostische verteilte Hashtabelle vorgestellt, welches keine virtuellen Knoten voraussetzt, und im Einklang mit aktuellen Sicherheitsmaßnahmen im Peer-to-Peer-Bereich umsetzbar ist. Im Zuge das nachfolgenden Abschnitts werden auch Kriterien für speicherplatzagnostische DHTs erarbeitet, welche insbesondere Sicherheitsaspekte berücksichtigen.

4. Kademlia für speicherplatzagnostische DHTs

Das hier vorgestellte DHT-Konzept basiert auf Kademlia und berücksichtigt darüber hinaus aktuelle Sicherheitsanforderungen und entsprechende Erkenntnisse der letzten Jahre. Insbesondere ist aus Sicherheitsgründen keine freie Identifikatorwahl gestattet. Das Kademlia-Routingprotokoll wurde nur gezielt angepasst, um speicherplatzagnostisches Verteilen von Daten zu ermöglichen. Gleichzeitig wurde darauf Wert gelegt, dass jeder Knoten unabhängig von seiner Speicherkapazität den gleichen Beitrag zum verteilen Routingprotokoll leistet. Als Ausgangspunkt hierfür wird nachfolgend das traditionelle Kademlia-Routingprotokoll beschrieben.

4.1. Ursprüngliches Kademlia-Routingprotokoll

Das Kademlia-Routingprotokoll setzt voraus, dass jeder Knoten über einen (im ursprünglichen Design selbst gewählten 160 Bit langen) Identifikator verfügt. Als Distanzfunktion kommt XOR zum Einsatz; die Distanz zwischen zwei Knoten berechnet sich daher wie folgt: $d(x, y) = ID_x \oplus ID_y$. Darüber hinaus sind zwei Parameter definiert:

- Replikationsfaktor k
- Parallelisierungsfaktor α

Die von jedem Knoten individuell verwalteten Routingtabellen bestehen aus so genannten *k-Buckets*. Beschreiben lässt sich eine solche Routingtabelle als Array von sortierten Listen. Ein Eintrag im Array am Index i beinhaltet eine Liste von Knoten (bzw. Abbildungen von Identifikatoren auf die Netzwerkadressen von Knoten) einer Distanz zwischen 2^i und 2^{i+1}. Der Name *k-Bucket* leitet sich davon ab, dass maximal k viele Knoten in einer Liste aufgenommen werden. Sortiert wird die Liste anhand des Zeitpunkts an dem Knoten zuletzt erreicht werden konnten.

Eine Routinganfrage wird folgendermaßen abgearbeitet: Die α nächsten Knoten zum Identifikator des gesuchten Knotens werden mit einer Anfrage, welche den Identifikatior des aufzufindenden Zielknotens enthält, kontaktiert. Jeder dieser Knoten antwortet mit den k dem Zielidentifikator nächstgelegenen ihm bekannten Knoten. Wenn alle Antworten eingegangen sind, wird daraus die Vereinigungsmenge gebildet und aus dieser wieder die α nächsten Knoten zum Identifikator des gesuchten Knotens mit einer Anfrage kontaktiert. Die Informationen zu bisher nicht bekannten Knoten werden in die Routingtabelle aufgenommen. Ist ein k-Bucket voll, so wird ein neuer Knoten nur dann hinzugefügt, wenn es einen Knoten im k-Bucket gibt, der auf eine Keep-Alive-Anfrage nicht antwortet. Dieser α-k-Frage-Antwort-Prozess wird bis zur Konvergenz wiederholt. Existiert der gesuchte Knoten im Netzwerk, konvergiert die Anfrage bei diesem Knoten. Ist dies nicht der Fall, konvergieren Anfragen an Knoten, die dem gesuchten Ziel am nächsten kommen. Ein Beweisskizze, welche erläutert, dass dieses Routingverfahren korrekt arbeitet, kann der ursprünglichen Publikation zu Kademlia [10] entnommen werden. Eine entscheidende Eigenschaft der gewählten Distanzfunktion und des an sich simplen Routingverfahrens ist Symmetrie: Routingtabellen können nicht nur auf Basis von Antworten auf Routinganfragen befüllt werden, sondern auch direkt auf Basis der Anfragen selbst (da diese Identifikator und Netzwerkadresse des Absenders enthalten). Grund dafür ist, dass die XOR-Distanz von *A* nach *B* gleich der Distanz von *B* nach *A* ist. Darüber hinaus erfüllt XOR die Dreiecksungleichung, wodurch Kademlia insgesamt einen metrischen Raum im mathematischen Sinn definiert. Dadurch lassen sich viele Gesetzmäßigkeiten, welche auch in euklidischen Räumen gelten ebenfalls auf Kademlia umlegen, wodurch Korrektheit und Vollständigkeit einfacher argumentierbar ist. Darüber hinaus ist ein intuitives Verständnis leichter zu erreichen.
Bevor die im Rahmen dieses Projekts entwickelte Erweiterung des Kademlia-Protokolls vorgestellt wird, werden grundlegende Anforderungen an Ansätze basierend auf angepassten Distanzfunktionen diskutiert.

4.2. Anforderungen an angepasste Distanzfunktionen

Aus der verteilten Natur von DHTs ergibt sich zwingend, dass einzelne Knoten immer im Verhältnis zur Gesamtkapazität des Netzwerks ausgelastet werden. Dies trifft sowohl auf den Anteil an Routingprozessen als auch auf die Häufigkeit, mit der ein Knoten zum Speichern von Daten ausgewählt wird, zu. Eine ungeschickte Anpassung der Distanzfunktion kann diese Verhältnismäßigkeit jedoch umkehren, wodurch einzelne Knoten mit wachsender Netzwerkgröße zunehmend an Einfluss gewinnen. Das folgende Beispiel illustriert diese Problematik ausgehend von XOR als Distanzfunktion.
Das generelle Ziel speicherplatzagnostischer Distanzfunktionen lässt sich wie folgt formulieren: Man will erreichen, dass bestimmte Knoten näher an allen möglichen Daten liegen. Zuständig für die Speicherung ist der Knoten, welche die Distanzfunktion $ID_{Data} \oplus ID_{Node}$ minimiert. Ohne den Einsatz virtueller Knoten kann dies durch das freie Belegen von Bits des Knoten-Identifikators erreicht

werden. Ein Knoten, welcher dem Netzwerk mehr Speicherplatz zu Verfügung stellen möchte, definiert wie viele Bits (ausgehend vom höchstwertigen, d.h. *most significant bit* MSB) in die Distanzberechnung einfließen sollen. Wenn beispielsweise die zwanzig niederwertigsten Bits nicht in die Berechnung der Distanz einfließen, ist die Chance, dass zufällig gewählte Daten in die Zuständigkeit dieses Knotens fallen, bezogen auf den Identifikatorraum um den Faktor 2^{20} höher. Allerdings muss dieser Wert noch mit der Belegung des Identfikatorraums (also wie viele Knoten es tatsächlich im Verhältnis zur Größe des Identifikatorraums gibt) abgeglichen werden. Hier zeigt sich die Problematik dieses Ansatzes: Um in einem kleinen Netzwerk von z.B. 100000 Knoten für mehr als durchschnittlich $1/100000$ aller Daten zuständig zu sein, sind lediglich die ersten $\log_2 1000000 \approx 20$ Bit des Identifikators eines Knotens entscheidend. Wenn das Ziel ist, im Durchschnitt für doppelt so viele Daten wie jeder beliebige andere Knoten verantwortlich zu sein, müssen alle bis auf die neunzehn höchstwertigen Bits bei der Distanzberechnung vernachlässigt werden. Verdoppelt sich die Netzwerkgröße, werden jedoch bereits 21 Bit bei der Distanzberechnung schlagend, wodurch sich der Einfluss der neunzehn höchstwertigen Bits verdoppelt.
Ein weiterer (naiver) Ansatz, Speicherkapazität in die Distanzfunktion einfließen zu lassen, wäre die errechnete Distanz mit einem speicherplatzabhängigen Faktor zu multiplizieren. Allerdings wird dadurch die Dreiecksungleichung verletzt, was dazu führt, dass Routingpfade über Knoten kürzer als direkte Pfade zu einem Zielknoten erscheinen können. Zusammenfassend lassen sich auf Grund dieser Beobachtung und unter Berücksichtigung aktueller Sicherheitsmaßnahmen folgende Kriterien für speicherplatzagnostische DHTs ableiten:

1. Knoten, welche mehr Speicherplatz zur Verfügung stellen, dürfen nicht mehr Einfluss auf das Routingprotokoll haben (siehe Sybil-Attacke [15] und Eclipse-Attacke [3]).
2. Der Anteil eines Knotens am Gesamtspeicherplatz des Netzwerks muss sich bei variabler Netzwerkgröße indirekt proportional dazu verhalten (siehe oben).
3. Routinganfragen müssen weiterhin in $\mathcal{O}(\log n)$ vielen Schritten bewältigbar sein (Prozesse im Netzwerk müssen in derselben Komplexitätsklasse bleiben).
4. Identifikatoren dürfen nicht frei wählbar sein (ebenfalls um diverse Angriffe, wie z.B. Eclipse-Attacken zu verhindern[2]).

Ansätze basierend virtuellen Knoten können die Anforderungen 1 und 3 prinzipbedingt nie erfüllen und scheiden daher als Alternative aus.
Nachfolgend wird eine Erweiterung des Kademlia-Protokolls vorgestellt, welche speicherplatzagnostisches Verteilen von Daten ermöglicht und gleichzeitig alle zuvor definierten Anforderungen erfüllt. Letzteres ist besonders deshalb relevant, da unter diesen Bedingungen alle bestehenden Sicherheitsmaßnahmen für strukturierte Peer-to-Peer-Netze und DHTs unverändert effektiv arbeiten, bzw. die Voraussetzungen für die Effektivität bestehender Maßnahmen weiterhin erfüllt bleiben.

4.3. Speicherplatzagnostisches Kademlia-Protokoll

Das im Rahmen dieses Projekts entwickelte speicherplatzagnostische Kademlia-Protokoll verwendet Ansätze zu virtuellen Knoten als Teil einer modifizierten Distanzfunktion, ohne dass dadurch der Einfluss einzelner Knoten auf das verteilte Routingprotokoll zunimmt. Jeder Knoten verfügt wie im ursprünglichen Protokoll über genau einen Identifikator. Zusätzlich wählt jeder Knoten abhängig vom zur Verfügung zu stellenden Speicherplatz einen Parameter, welcher angibt, über wie viele virtuelle Identifikatoren er zusätzlich verfügt. Da ausgehend von Anforderung 4 Identifikatoren nicht frei wählbar sein dürfen, handelt es sich dabei lediglich um eine Zahl s, nicht jedoch um konkrete Identifikatoren. Die Identifikatoren selbst werden durch die Anwendung einer kryptografischen Hashfunktion H wie folgt berechnet: $ID_i = H(ID_{Node}||i)\ \forall i \in [0..s)$. Wichtig ist in diesem Zusammenhang, dass die Hashfunktion auf ebensoviele Bits abbildet, wie Identifikatoren von Knoten und Daten und dass eine logische Trennung zwischen tatsächlichen Identifikatoren und virtuellen Identifikatoren vorgenommen wird.

[2] https://technology.a-sit.at/sichere-peer-to-peer-netze-auf-basis-von-selbstzertifizierung/

Reguläre Routinganfragen werden nach wie vor auf tatsächliche Identifikatoren angewandt. Um festzustellen, welcher Knoten für die Speicherung von Daten zuständig ist, wird hingegen die Distanz zwischen Daten-Identifikator und virtuellen Identifikatoren berechnet. Eine Anfrage nach für ein Datum zuständige Knoten liefert allerdings nach wie vor k viele Knoten (und nicht k viele virtuelle Identifikatoren; siehe Abschnitt 4.1), um zu verhindern, dass Knoten mit vielen virtuellen Identifikatoren mehr Einfluss auf das verteilte Routingprotokoll haben (vgl. Kriterium 1). Die auf eine Anfrage retournierten Knoten sind im Gegensatz zu Routinganfragen jedoch auf Basis der Distanzen ihrer virtuellen Identifikatoren zum Daten-Identifikator sortiert. Statistisch gesehen sind Knoten mit mehr virtuellen Identifikatoren öfter für die Speicherung von Daten zuständig als Knoten mit weniger Identifikatoren. Damit dies auch tatsächlich mit dem verursachten Speicherplatzverbrauch korrespondiert, müssen die zu speichernden Daten annähernd gleich groß sein. Dies kann beispielsweise durch eine gleichmäßige Aufteilung großer Daten und anschließender Organisation als Merkle-Baum [17] erfolgen. In diesem Zusammenhang ist es darüber hinaus sinnvoll, netzwerkweit festzulegen welcher kleinsten Speichereinheit ein virtueller Identifikator entspricht, bzw. der durchschnittlichen Speicherkapazität eines Knotens eine Anzahl virtueller Identifikatoren zuzuweisen, sodass sich die Anzahl virtueller Identifikatoren in sinnvollen Grenzen hält.
Die Komplexitätsklasse von Routinganfragen bleibt mit $\mathcal{O}(\log n)$ unverändert. Der benötigte Aufwand, um für Daten zuständige Knoten aufzufinden, erhöht sich um einen konstanten Faktor, der der durchschnittlichen Anzahl virtueller Identifikatoren pro Knoten entspricht. Die Anzahl an Hops, die benötigt werden, um Daten oder Knoten zu lokalisieren, beträgt ebenfalls nach wie vor $\mathcal{O}(\log n)$. Des Weiteren gibt es dadurch keine unerwünschten Nebeneffekte (beispielswiese definiert das Routingprotokoll nach wie vor einen metrischen Raum). Insgesamt erfüllt dieses Protokoll alle geforderten Kriterien, wodurch bestehende Peer-to-Peer-Sicherheitskonzepte angewandt werden können. Details hierzu werden im nachfolgenden Abschnitt diskutiert. Ein im Rahmen dieses Projekts entwickelter Demonstrator veranschaulicht darüber hinaus die Funktionalität des Protokolls.

4.4. Sicherheitsaspekte

Einen signifikanten Beitrag zu Peer-to-Peer-Sicherheit leisten selbstzertifizierende Identifikatoren. Knoten-Identifikatoren können im Rahmen dieses Konzepts nicht mehr frei gewählt werden, sondern werden vom öffentlichen Teil eines asymmetrischen Schlüsselpaars abgeleitet. Zusätzlich müssen alle im Netzwerk versendeten Nachrichten vom Absender mit dem zum Identifikator passenden privaten Schlüssel signiert werden. Darüber hinaus müssen Routingtabellen dahingehend erweitert werden, dass Nachrichten (welche Absenderinformationen enthalten müssen) im Ganzen gesichert werden, um signierte Informationen weitergeben zu können. Dadurch lässt sich einerseits die Quelle aller im Netzwerk verteilten Informationen feststellen und andererseits sicherstellen, dass ausschließlich integre Informationen verbreitet werden. Das entwickelte Konzept ist mit diesem Ansatz kompatibel, da das verwendete Nachrichtenformat lediglich um ein Feld erweitert werden muss, welches angibt, über wie viele virtuelle Identifikatoren ein Knoten verfügt. Auf eine eventuelle Zertifizierung von Knoten-Identifikatoren im Rahmen von PKIs wird dadurch ebensowenig beeinträchtigt. Da der Wert der virtuellen Identifikatoren nicht gezielt gewählt werden kann, und Routing nach wie vor auf Basis der tatsächlichen Identifikatoren stattfindet, können auch Maßnahmen gegen Sybil- und Eclipse-Attacken unverändert angewandt werden. Auch (Sicherheits-) Konzepte bezüglich redundantem Routing können unverändert umgesetzt werden, da der Einsatz von virtuellen Identifikatoren keinen Einfluss darauf hat, wie viele Knoten an einer Routinganfrage beteiligt sind. Sichere Übertragung von Informationen kann durch den Einsatz von *Datagram Transport Layer Security* (DTLS) ebenfalls umgesetzt werden. In Kombination mit selbstzertifizierten Identifikatoren erübrigt sich auch die Frage des Schlüsselmanagements, da das vorhandene Schlüsselmaterial benutzt werden kann, um jeden Knoten zweifelsfrei zu identifizieren. Für den Fall, dass Daten nur einer bestimmten Menge von Knoten zugänglich gemacht werden sollen, können diese (z.B. als CMS-Container) auf Basis selbstzertifizierender Identifikatoren verschlüsselt in der DHT gespeichert werden.

5. Fazit

Im Rahmen dieses Projekts wurden Ansätze zur Umsetzung speicherplatzagnostischer verteilter Netzwerkspeicher dargelegt und ein Konzept für eine speicherplatzagnostische verteilte Hashtabelle

entwickelt. Dabei werden Ansätze basierend auf virtuellen Knoten und Anpassungen der verwendeten Distanzfunktion kombiniert, um eine Verteilung basierend auf der Speicherkapazität der Knoten zu erreichen, ohne jedoch andererseits die Bedingungen für den Einsatz verschiedener Sicherheitsmaßnahmen zu verletzen. Ein Demonstrator veranschaulicht die Umsetzbarkeit des vorgestellten Konzepts.
Das entwickelte Konzept für eine speicherplatzagnostische verteilte Hashtabelle kann in weiterer Folge als Teil eines Netzwerkspeicherkonzepts angewandt werden, um einen verteilten Netzwerkspeicher umzusetzen, welcher einerseits die Speicherplatzheterogenität im Netzwerk berücksichtigt und über den Einsatz von lokalen Caches und entsprechender Cache-Replacement-Strategien aktuell benötigte Daten auf allen Geräten repliziert. Dadurch kann erreicht werden, dass im Rahmen von Arbeitsabläufen, die zeitlich auf mehrere Geräte verteilt ausgeführt werden, zu jedem Zeitpunkt aktuell benötigte Daten überall lokal verfügbar sind.

Referenzen

[1] B. Cohen, „The BitTorrent Protocol Specification," 11 10 2013. [Online]. Available: http://www.bittorrent.org/beps/bep_0003.html. [Zugriff am 24 04 2018].

[2] I. Baumgart und S. Mies, „S/Kademlia: A practicable approach towards secure key-based routing," in *2007 International Conference on Parallel and Distributed Systems*, 2007.

[3] Atul Singh et al., „Defending Against Eclipse Attacks on Overlay Networks," in *Proceedings of the 11th Workshop on ACM SIGOPS European Workshop*, Leuven, ACM, 2004.

[4] R. Fantacci et al., „Avoiding Eclipse Attacks on Kad/Kademlia: An Identity Based Approach," in *2009 IEEE International Conference on Communications*, IEEE, 2009.

[5] Haifeng Yu et al., „SybilGuard: Defending Against Sybil Attacks via Social Networks," in *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Pisa, ACM, 2006, pp. 267-278.

[6] B. N. Levine, C. Shields und N. B. Margolin, „A Survey of Solutions to the Sybil Attack," Amherst, 2006.

[7] Frank Li et al., „SybilControl: Practical Sybil Defense with Computational Puzzles," in *Proceedings of the Seventh ACM Workshop on Scalable Trusted Computing*, Raleigh, ACM, 2012, pp. 67-78.

[8] Bimal Viswanath et al., „An Analysis of Social Network-based Sybil Defenses," in *Proceedings of the ACM SIGCOMM 2010 Conference*, Neu Delhi, ACM, 2010, pp. 363-374.

[9] B. Prünster, „Sichere Peer-to-Peer-Netze auf Basis von Selbstzertifizierung," 07 05 2018. [Online]. Available: https://technology.a-sit.at/sichere-peer-to-peer-netze-auf-basis-von-selbstzertifizierung/. [Zugriff am 11 07 2018].

[10] P. Maymounkov und D. Mazières, „Kademlia: A Peer-to-Peer Information System Based on the XOR Metric," in *Peer-to-Peer Systems*, Springer, 2002, pp. 53-65.

[11] Ion Stoica et al., „Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," in *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, San Diego, ACM, 2001, pp. 149-160.

[12] Sylvia Ratnasamy et al, „A Scalable Content-addressable Network," in *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, San Diego, ACM, 2001, pp. 161-172.

[13] A. Rao, K. Lakshminarayanan, S. Surana, R. Karp und I. Stoica, „Load Balancing in Structured P2P Systems," in *Peer-to-Peer Systems II*, Springer, 2003, pp. 68-79.

[14] P. Godfrey und I. Stoica, „Heterogeneity and Load Balance in Distributed Hash Tables," in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, IEEE, 2005, pp. 596-606.

[15] J. R. Douceur, „The Sybil Attack," in *Peer-to-Peer Systems*, Springer, 2002, pp. 251-260.

[16] S. Kniesburges, A. Koutsopoulos und C. Scheideler, „CONE-DHT: A Distributed Self-Stabilizing Algorithm for a Heterogeneous Storage System," in *International Symposium on Distributed Computing*, Springer, 2013, pp. 537-549.

[17] R. C. Merkle, „A Digital Signature Based on a Conventional Encryption Function," in *Advances in Cryptology — CRYPTO '87*, Springer, 1987, pp. 369-378.