

Content Delivery Networks und deren Funktionsweise

Christoph Riesinger

Universität Passau, 94030 Passau, Deutschland

Zusammenfassung Die stetig steigende Zahl von Benutzern des Internet, insbesondere des World Wide Web, führt zu einer ansteigenden Last der Server, die Inhalte, Dienste und Dateien zur Verfügung stellen. Diese erhöhte Last kann dazu führen, dass Server überlastet werden. Das kann zur Folge haben, dass Dienste nicht mehr zufriedenstellend in Bezug auf Antwortzeiten, Bandbreite, Fehlerfreiheit, Verfügbarkeit und/oder Vollständigkeit angeboten werden können.

Content Delivery Networks bieten eine Lösung für dieses Problem. Dieser Text stellt Content Delivery Networks vor, demonstriert deren Funktionsweise und geht näher auf Algorithmen ein, die bei Content Delivery Networks auftretende Herausforderungen bewältigen.

1 Einführung

Zu Beginn dieser Arbeit soll eine Einführung an das Thema *Content Delivery Networks* (kurz: CDN) heranzuführen. Sie zeigt die Notwendigkeit von CDN auf und skizziert deren grundlegende Funktionsweise. Dabei wird auch bereits auf die größten Herausforderungen bei CDN eingegangen.

1.1 Limitierungen klassischer Netze

In Zeiten der Konzeption des Internet und später des *World Wide Web* (kurz: WWW) wurde bewusst eine Client/Server-Struktur gewählt: Einzelne wenige Server sollten einer Vielzahl von Benutzern Dienste zur Verfügung stellen. Dieser Ansatz birgt jedoch diverse Nachteile. Einige davon seien an dieser Stelle erwähnt:

- Eine zu große Benutzerzahl kann die Serverseite verlangsamen oder sogar überlasten, so dass der Server seine Aufgabe nicht mehr erfüllen kann.
- Latenzzeiten können sehr groß sein, da die Möglichkeit besteht, dass sich Client und Server an zwei vollkommen unterschiedlichen Orten auf der Erde befinden. Ebenso steigt die Wahrscheinlichkeit eines Flaschenhalses auf der Route zwischen Client und Server, da man davon ausgehen kann, dass je weiter Server und Client voneinander entfernt sind, mehr Zwischenstationen im Netzwerk passiert werden müssen. Jede dieser Zwischenstationen beziehungsweise jede der Leitungen zwischen den Stationen kann die Qualität der Verbindung verschlechtern.
- Fällt ein Server komplett aus, so sind auch dessen geleistete Dienste nicht mehr erreichbar. Ein Ersatzsystem ist normalerweise nicht vorgesehen.

1.2 Lösung der Limitierungen: CDN leitet Anfragen zu Kopien

Um die oben erwähnten Probleme zu lösen, verwendet man Content Delivery Networks. Diese funktionieren wie folgt: Anstatt eine Datei oder einen Dienst nur auf einem einzelnen Server bereit zu stellen, werden zahlreiche Kopien dieser Datei oder dieses Dienstes auf mehrere Server im Netzwerk oder im Internet verteilt. Die Last der Nachfrage kann nun auf diese Kopieserver (auch Mirrors, Spiegelserver, Center oder Replica Server) aufgeteilt werden, so dass sich die Last in Folge verteilt. Dies führt zu einer geringeren Auslastung eines jeden einzelnen Kopieserver.

1.3 Herausforderungen an ein CDN

Diese Arbeit soll einen Überblick über die Funktionsweise von CDN bieten. Dabei stehen zwei zentrale Anforderungen eines CDN im Mittelpunkt:

- Bei der ersten Herausforderung handelt es sich um die Frage, wie eine gewisse Anzahl von Kopien auf eine konkrete Anzahl von Mirrors verteilt werden kann, so dass für jeden Benutzer die Bandbreite zu dem Server optimal, die Zugriffszeit minimal und die Verfügbarkeit maximal ist. Mit dieser Frage beschäftigt sich Kapitel 4.
- Als zweite Herausforderung gilt es die Frage zu lösen, wie für einen konkreten Benutzer, der für ihn beste Kopieserver ermittelt wird und wie eine Umleitung seiner Anfrage nach dem geforderten Inhalt an den eben ermittelten optimalen Server erfolgen kann. Punkt 5 klärt diese Frage.

2 Dienste und Leistungen eines CDN

Bevor auf die Bewältigung der Herausforderungen eines CDN eingegangen wird, soll an dieser Stelle kurz skizziert werden, welche Dienste und Leistungen ein CDN bietet.

Wie bereits erwähnt, dient ein CDN dazu einen WWW-Dienst optimal im Sinne von hoher Bandbreite zum Server, minimaler Zugriffszeit und maximaler Verfügbarkeit zugänglich zu machen. Diese WWW-Dienste können unterschiedlichster Natur sein. Grob kann man sie in drei Kategorien unterteilen (in Anlehnung an [1]):

- **Statische Dateidienste.** Hierbei sollen sich selten verändernde Dateien Benutzern zugänglich gemacht werden. Diese Dateien können unterschiedlich groß sein und unterschiedlich oft frequentiert werden. Sie reichen von wenigen Kilobyte großen HTML-Dateien über ein paar Megabyte große Softwareupdates bis hin zu Gigabyte großen Softwaredistributionen wie Betriebssystemen.
- **Anwendungen.** In diese Kategorie fallen Onlineshops wie Amazon¹ oder eBay² genauso wie die im Moment aufkommenden Liveanwendungen wie

¹ Homepage von Amazon: <http://www.amazon.de/>

² Homepage von eBay: <http://www.ebay.de/>

Microsofts OfficeLive³ oder Social Networking Sites wie MySpace⁴. Zumeist arbeiten im Hintergrund solcher Angebote weitere Anwendungen wie Datenbanken.

- **Live oder On-Demand Streaming.** Bei dieser Kategorie handelt es sich um die Distributoren von Multimediainhalten. Charakteristisch ist die enorme Größe der zu transferierenden Inhalte. Auch das Kriterium „Live“ bedarf besonderer Handhabung. Anbieter solcher Inhalte reichen von Streamingseiten wie YouTube⁵ über On-Demand Portale wie Maxdome⁶ bis hin zur NASA⁷, die zahlreiche Live Videostreams zu ihren Missionen anbietet.

Ein CDN muss flexibel genug sein, um jeden Dienst aus den obigen drei Kategorien anzubieten und dabei auf die jeweiligen Besonderheiten eingehen können. Ein solcher Anbieter für CDN ist Akamai⁸, der weltweit wohl bekannteste Anbieter von CDN.

3 Metriken zur Bestimmung des „besten Server“

In dieser Arbeit ist mehrmals die Rede von einem für einen Client „besten Spiegelserver“ oder „optimalen Spiegelserver“. Jedoch gibt es unterschiedliche Kriterien und Metriken, wie man für einen Client denjenigen Kopieserver bestimmt, der die Anfragen eines Client am besten bedient. Drei dieser Metriken sollen in diesem Abschnitt vorgestellt werden. Alternative und für ein konkretes Szenario besser geeignete Metriken sind denkbar. Eine universell beste Metrik gibt es nicht, denn je nach Anwendung müssen unterschiedliche Kriterien der Bewertung heran gezogen werden.

3.1 Geographische Nähe

Bei dieser Metrik ist ein Server umso besser für einen Client, je näher er geographisch, also in der realen Welt, zu ihm liegt. Server, die in derselben Region oder im selben Land liegen wie der anfragende Client werden als besser erachtet als Server, die auf einem anderen Kontinent liegen.

Der Vorteil dieser Metrik besteht darin, dass es sich um statische Werte handelt, die einmalig bestimmt werden können. Schließlich sind Clients und Server in der Regel stationär, ändern also ihre Position nicht wesentlich.

Jedoch muss man anmerken, dass diese Metrik keinerlei Informationen über die tatsächliche Netzwerkinfrastruktur zwischen Client und Server beachtet.

³ Homepage von OfficeLive: <http://office.microsoft.com/de-de/officelive/default.aspx>

⁴ Homepage von MySpace: <http://www.myspace.com/>

⁵ Homepage von YouTube: <http://www.youtube.com/>

⁶ Homepage von Maxdome: <http://www.maxdome.de/>

⁷ Homepage der NASA: <http://www.nasa.gov/>

⁸ Homepage von Akamai: <http://www.akamai.com/>

3.2 Anzahl der Zwischenstationen

Im Internet erreichen Nachrichten ihr Ziel über diverse Zwischenstationen, sogenannte *Router*, die die zu versendenden Nachrichten in Subnetze weiterleiten. Jedoch können an diesen Routern Latenzzeiten auftreten, beispielsweise hervorgerufen durch eine hohe Last des Routers oder wenn die Route zum Ziel der Nachricht erst noch berechnet werden muss.

Man könnte nun sagen, dass derjenige Server für einen Client der Beste ist, der am wenigsten Zwischenstationen weit von ihm entfernt ist.

Dieses Kriterium liefert jedoch nicht immer das beste Ergebnis: So ist es denkbar, dass die gesamte Latenzzeit der Router einer Route mit wenigen Zwischenstationen größer ist, als die einer Route mit vielen Zwischenstationen. Dies könnte beispielsweise der Fall sein, wenn die wenigen Zwischenstationen allesamt eine hohe Last haben.

Darüber hinaus wird bei dieser Metrik nur die Latenzzeit der Zwischenstationen beachtet, nicht jedoch die der Leitungen. So wird bei einer transatlantischen Verbindung eine Strecke von über 4000 km zurück gelegt, obwohl dabei nur eine einzige Zwischenstation passiert wird. Die Latenzzeit dieser langen Leitung liegt jedoch aufgrund ihrer Länge im zweistelligen Millisekundenbereich.

Allerdings hat man hier wiederum den Vorteil, dass die Anzahl der Zwischenstationen zumeist statisch ist. Zwar erlaubt es das *Internet Protocol* (kurz: IP) während einer Sitzung eine alternative Route zu wählen, in der Praxis jedoch findet dies keine Verwendung, da die Berechnung einer alternativen Route viel zu aufwändig ist und daher eher auf statische Einträge in einer Routingtabelle zurück gegriffen wird.

3.3 Latenzzeit und Antwortzeit

Die wohl aussagekräftigste Metrik ist die der *Latenzzeit* beziehungsweise die der *Antwortzeit*. Bei der Latenzzeit wird die Zeit gemessen die vergeht, bis eine bei der Quelle auf die Leitung gelegte Nachricht ihr Ziel erreicht. Die Antwortzeit gibt die Zeit an die vergeht, bis ein Client eine Antwort auf seine Anfrage erhält, die er an einen konkreten Server gestellt hat.

Zwar sagt die Latenzzeit lediglich etwas über die Qualität der Leitung zum Server, nicht jedoch über den Server selbst aus, die Antwortzeit liefert jedoch sowohl eine qualitative Aussage über die Verbindung als auch eine über den Server selbst.

Größter Nachteil dieses Verfahrens ist allerdings, dass es sich bei diesen beiden Zeiten um hoch dynamische Werte handelt, die sich schnell ändern können, weshalb sie regelmäßig aktualisiert werden müssen. Dies erzeugt Overhead.

4 Kopieplatzierung

Dieser Abschnitt beschäftigt sich mit der Aufgabe eines CDN, Kopien von Dateien auf Spiegelserver zu verteilen.

Die hier zu lösende Aufgabe besteht darin, eine optimale Teilmenge von potentiellen Replica Servern zu bestimmen, so dass die Benutzer optimalen Zugriff auf die angebotenen Dateien haben. H bezeichne in Folge die Menge der potentiellen Replica Server, M die Menge der schlussendlich ausgewählten und eingesetzten Server. $|M| = K$, wobei K in der Regel als Kriterium gegeben wird. Normalerweise stehen mehr potentielle Server zur Verfügung als dann in der Praxis eingesetzt werden, also gilt $M \subseteq H$. Alle Benutzer, die einen Dienst anfordern, sind in der Menge B zusammen gefasst.

Dieses Problem ist NP-vollständig, jedoch existieren in der Praxis Algorithmen, deren Laufzeit einen Praxiseinsatz erlaubt. Zwei von ihnen werden an dieser Stelle erläutert.

Handelt es sich um Ressourcen, die über das Internet zugänglich gemacht werden sollen, so sind diese potentiellen Server eines CDN normalerweise weltweit verteilt.

4.1 Theoretische Grundlagen

Um die in Folge erläuterten Algorithmen zu begreifen, seien an dieser Stelle ein paar Begriffe aus dem Bereich von Graphen erwähnt.

Ein *Graph* $G = (V, E)$ besteht aus einer *Knotenmenge* V und einer *Kantenmenge* $E \subseteq V \times V$. Eine *Kostenfunktion* $c(e)$ gibt die Kosten der Kante zwischen den Knoten v_1 und v_2 an, wobei $e = (v_1, v_2)$. Übertragen auf ein Netzwerk ist V die Menge aller Rechner im Netzwerk und E spezifiziert die Verbindungen unter allen Rechnern.

Sortiert man die Kanten eines Graphen nach ihren Kosten, $c : c(e_1) \leq c(e_2) \leq \dots \leq c(e_m)$, so ist $G_i = (V, E_i)$ der Graph mit gleicher Knotenmenge V und modifizierter Kantenmenge $E_i = \{e_1, e_2, \dots, e_i\}$, also den i Kanten mit dem geringsten Kantengewicht.

Ein *Squaregraph* G^2 von G ist ein Graph, der dieselbe Knotenmenge V wie G besitzt, jedoch zusätzlich Kanten (u, v) , bei denen gilt, dass der Knoten v vom Knoten u aus in G in höchstens zwei Sprüngen erreichbar ist, $u \neq v$. Abbildung 1 zeigt, welche Kanten (rot) zur ursprünglichen Kantenmenge (schwarz) hinzugefügt werden, wenn zu einem gegebenen Graphen G dessen Squaregraph G^2 bestimmt werden soll.

Eine *unabhängige Menge* eines Graphen $G = (V, E)$ ist eine Knotenteilmenge $V' \subseteq V$, so dass für alle $u, v \in V'$ die Kante (u, v) nicht in E existiert. Das bedeutet also, dass zwischen den Knoten in V' keine Kanten verlaufen. Somit lässt sich auch eine *maximale unabhängige Menge* V'_m definieren: Es lassen sich keine weiteren Knoten $w \in V$ in V'_m einordnen, so dass $\{w\} \cup V'_m$ eine unabhängige Menge ist. Grafik 2 zeigt sowohl eine unabhängige als auch eine maximal unabhängige Knotenmenge.

4.2 Greedy Algorithmus

Dieser *Greedy Algorithmus* [2] arbeitet wie jeder Greedy Algorithmus, indem er fortlaufend ein lokales Optimum bestimmt und anschließend diese lokalen Opti-

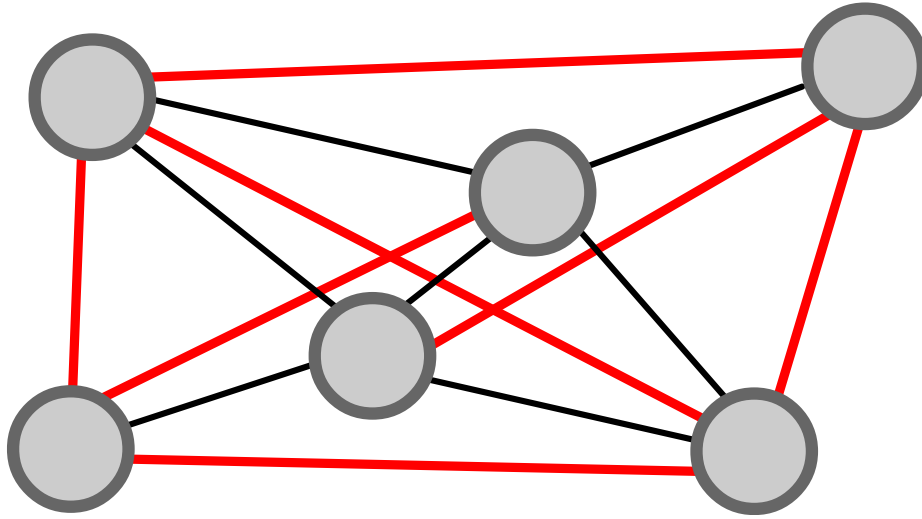


Abbildung 1: Diese Abbildung zeigt einen beispielhaften Graphen (Kanten: schwarz, Knoten: grau) G und die Kanten (rot), die bei der Bestimmung des Squaregraphen G^2 hinzugefügt werden.

ma zu einer relativ guten Gesamtlösung vereint.

Im Falle von CDN bedeutet dies, dass im ersten Schritt derjenige Server aus der Menge der potentiellen Spiegelserver H ausgewählt wird, der für alle Clients die besten Konditionen bietet. Dieser optimale Server wird aus der Menge der potentiellen Spiegelserver H entfernt und in die anfangs leere Menge der Server M eingefügt, die später tatsächlich Spiegelserver werden. Aus der in ihrer Mächtigkeit um eins reduzierten Menge H wird nun wiederum der jetzt beste Server ausgewählt, der für alle Clients die besten Konditionen bietet. Dieser wird wie im ersten Schritt aus der Menge H entfernt und in die Menge M eingefügt. Dieses Verfahren wird so lange wiederholt, bis man K Spiegelserver in der Menge M beisammen hat. Es handelt sich also um ein iteratives Verfahren.

4.3 min K-center Algorithmus

Wie bereits erwähnt, ist das Problem der Serverauswahl für die Kopien NP-vollständig. Der *min K-center Algorithmus* [3] liefert ein Ergebnis, bei dem jeder Client maximal doppelt so weit von seinem optimalen Server entfernt ist, wie beim globalen Optimum. Dieser Algorithmus arbeitet wie folgt:

1. Bilde die Squaregraphen $G_1^2, G_2^2, \dots, G_m^2$, $m = |E|$.
2. Berechne die maximalen unabhängigen Mengen V'_{mi} für jeden Graphen G_i^2 .
3. Finde das kleinste i , so dass $|V'_{mi}| \leq K$.
4. Für dieses kleinste i ist V'_{mi} die Menge M der für die Praxis auszuwählenden Spiegelserver.

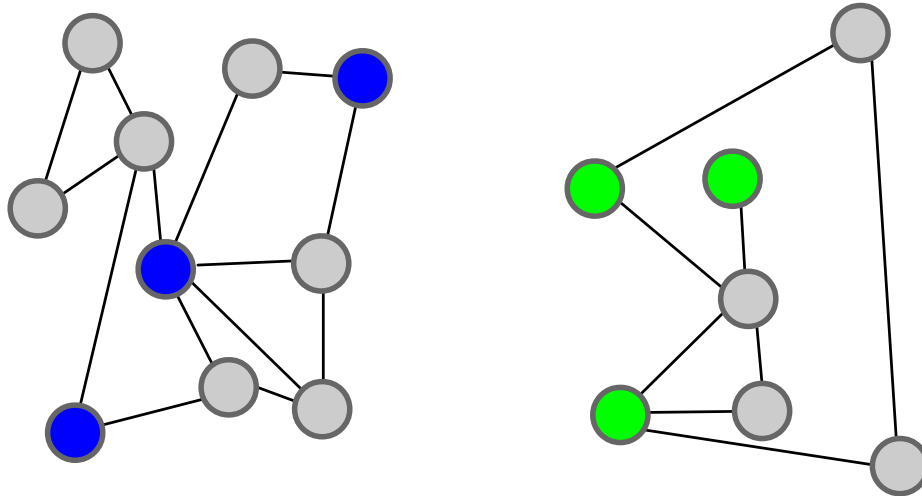


Abbildung 2: Die blauen Knoten des linken ungerichteten Graphen bilden eine unabhängige Menge V' . Die grünen Knoten des rechten ungerichteten Graphen bilden eine maximale unabhängige Menge V'_m .

Grafik 3 illustriert den min K-center Algorithmus. Zu beachten ist, dass bei diesem Algorithmus jeder Knoten des Graphen beziehungsweise jeder Rechner des Netzwerks als Server ausgewählt werden kann. Jedoch sollen nur Rechner aus der potentiellen Replica Server Menge H verwendet werden. Um dies zu erreichen, wird der Algorithmus zuerst auf einem Graphen mit $V = H \cup B$ angewandt. Sollte ein Knoten aus B als ein Spiegelserver ausgewählt werden, so wird er durch den ihm nächsten Knoten in H ersetzt.

4.4 Fazit

Abschließend bleibt zu sagen, dass beide Algorithmen von der Annahme ausgehen, das Verhalten der Benutzer zu kennen, die auf die verteilten Dateien zugreifen wollen. Beim Greedy Algorithmus ist dies die Kenntnis, welcher Server der beste für einen Client ist, beim min K-center Algorithmus ist dies die Kostenfunktion c , die die Verbindungsqualität zwischen den einzelnen Knoten beschreibt. In der Realität kann darüber natürlich keine verlässliche Aussage gemacht werden, zumindest nicht im Voraus. An dieser Stelle muss auf Erfahrungswerte und eventuell auf stochastische Modelle zurück gegriffen werden.

5 Serverauswahl und Anfrageumleitung

Nachdem die Verteilung der Kopien erledigt ist, bleibt nun die Frage zu klären, wie ein Benutzer zu der für ihn am besten geeigneten Kopie geleitet wird. Dies sollte möglichst transparent geschehen. Das bedeutet, dass der Benutzer von

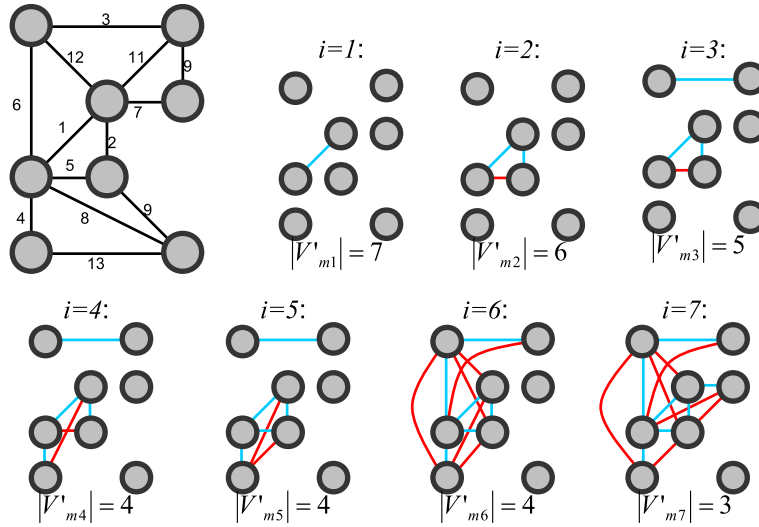


Abbildung 3: Der min K-center Algorithmus wird auf den gegebenen gewichteten Graphen links oben, welcher eine abstrakte Sicht auf ein Netzwerk darstellen soll, angewandt.

In diesem Beispiel läuft der Algorithmus nur bis $i = 7$, die Grafik zeigt die Squaregraphen G_1^2 bis G_7^2 .

Dabei sind jeweils mit türkis die i Kanten mit dem geringsten Kantengewicht eingezeichnet, mit rot die Kanten, die durch die Bestimmung des Squaregraphen von G_i^2 hinzu kommen. Die grün eingefärbten Knoten der Squaregraphen G_i^2 sind eine mögliche maximal unabhängige Menge V'_{mi} . $|V'_{mi}|$ ist die Mächtigkeit dieser maximal unabhängigen Menge.

Je nachdem welches K man gewählt hat, schaut man nun, ab welchem i die Ungleichung $|V'_{mi}| \leq K$ erstmals erfüllt ist und verwendet dieses V'_{mi} als Menge M .

Bei einer vollständigen Ausführung des Algorithmus müsste dieser bis $i = |E|$, also bis Anzahl der Kanten des Graphen laufen.

diesem Auswahlvorgang und der anschließenden automatischen Umleitung zum besten Server nichts mitbekommt. Für ihn soll es so aussehen, als würde er sich auf dem Originalserver bewegen.

In diesem Zusammenhang gilt es zwei Fragen zu klären: Wie wird der beste Server für einen Benutzer bestimmt und wie wird anschließend die Anfrage des Benutzers transparent zum besten Server umgeleitet.

5.1 Algorithmen zur Auswahl des besten Servers

Mehmet Sayal et al. [5] stellen zwei Algorithmen zur Auswahl des besten Replica Server für einen Client vor.

Wahrscheinlichkeitstheoretischer Algorithmus. Man geht von der Annahme aus, dass für jeden Client bekannt ist, welcher Replica Server für ihn der Beste

ist. Ebenso sei bekannt, welche Server weniger optimal für ihn sind. Die meisten Anfragen des Clients werden an diesen optimalen Server gerichtet. Von Zeit zu Zeit jedoch stellt der Client ein paar seiner Anfragen an einen weniger gut für ihn geeigneten Server. Diese Anfragen an weniger gut passende Server geschehen mit einer Häufigkeit, die indirekt proportional zur Antwortzeit des nicht optimalen Servers ist.

Nachteil dieses Verfahrens ist, dass man anfänglich nicht davon ausgehen kann, Wissen über die Güte der einzelnen Replica Server für den Client zu haben. Außerdem kann die zwischenzeitliche Wahl eines nicht optimalen Servers zu großen Wartezeiten und Einbußen beim *Quality of Service* (kurz: QoS) führen.

Im Gegensatz dazu kann dieser Algorithmus auf Veränderungen bei den Antwortzeiten der Server reagieren und das ganz ohne zusätzlichen Anfrageaufwand. Stellt sich bei den gelegentlichen Anfragen an die nicht optimalen Server heraus, dass die Antwortzeit zu einem dieser Server geringer ist als die zum bisherigen optimalen Server, so wird derjenige neuer optimaler Server, der die bessere Antwortzeit besitzt.

Aktualisierender Algorithmus. Bei diesem Algorithmus geht man wieder von der Annahme aus, dass für jeden Client bekannt ist, welcher Replica Server für ihn der Beste ist. Jedoch werden im Gegensatz zum Wahrscheinlichkeitstheoretischen Algorithmus alle Anfragen eines Clients an den für ihn besten Replica Server geleitet. Dieser momentan beste Replica Server ist allerdings mit einer Gültigkeitsperiode versehen, nach deren Ablauf aktiv geprüft wird, ob es sich beim aktuell ausgewählten Replica Server tatsächlich noch um den besten Server handelt. Dies kann beispielsweise mit einer asynchronen HTTP HEAD Anfrage geschehen.

Der Vorteil dieser Variante ist, dass im Gegensatz zum Wahrscheinlichkeitstheoretischen Algorithmus nicht der Fall eintreten kann, dass Anfragen an einen sehr schlechten Server gerichtet werden. Dies erkauft man sich mit dem Umstand, dass aktiv ermittelt werden muss, welcher Server gerade der beste für einen Client ist, was beim ersten Verfahren implizit geschieht.

Wiederum geht man auch hier davon aus, dass man zu Beginn weiß, welcher Server der optimale ist.

5.2 Methoden der Anfrageumleitung zum richtigen Spiegelserver

Nachdem nun der zu einem Client am besten passende Server gefunden wurde, wird nun die Frage geklärt, wie die Anfragen des Clients an diesen Server umgeleitet werden. Dies soll für den Client möglichst transparent geschehen, das bedeutet, dass der Client von dieser Anfrageumleitung nichts bemerken soll. Im Folgenden werden hierfür vier Methoden [4] vorgestellt.

Client basiert. Bei der Client basierten Serverauswahl handelt es sich um die einfachste Art der dynamischen Serverauswahl. Das Clientprogramm selbst

(beim Surfen im WWW wäre dies beispielsweise der Webbrowser) leitet die Anfrage an den korrekten Server, sobald ein Benutzer eine Datei oder eine Website aufruft.

Jedoch ist hierfür zwingend notwendig, dass die Clientsoftware die zur Auswahl stehenden Server kennt oder zumindest Informationen benötigt, welcher Server momentan für die angeforderte Datei der Beste ist. Dies sind tiefgehende Informationen über das CDN.

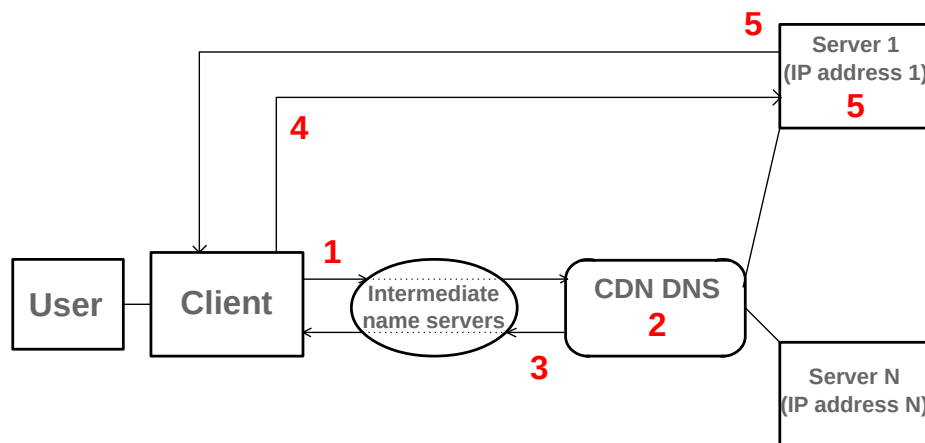


Abbildung 4: **DNS basierter Ansatz**. Schritt 1: Anfrage zur Domainname-IP-Adresse-Auflösung. Schritt 2: Auswahl eines Servers (hier Server 1). Schritt 3: Adressabbildung (URL → Adresse 1). Schritt 4: Dokumentanfrage an Server 1. Schritt 5: Server 1 sendet das angeforderte Dokument. Abbildung aus [4].

DNS basiert. *Domain Name Service* (kurz: DNS) Server sind Server, die einen gegebenen Domainnamen in die IP-Adresse des entsprechenden Server auflösen. Dieser Mechanismus kann für transparente Anfrageumleitung verwendet werden. Hierfür denke man sich folgendes Szenario (Abbildung 4 aus [4]): Ein CDN hält diverse Kopien eines Dienstes oder einer Datei bereit. Ein einzelner DNS Server übernimmt die Domainname-IP-Adresse-Auflösung für dieses CDN. Frägt nun ein Client eine konkrete Datei ab, die das CDN zur Verfügung stellt, so muss der in der Anfrage enthaltene Domainname zunächst vom DNS Server aufgelöst werden. Hierbei sendet der DNS Server diejenige IP-Adresse eines Kopieservers an den Client zurück, der für den Client momentan der Beste ist.

Der Vorteil dieses Verfahrens liegt darin, dass der Client keinerlei Kenntnisse über das CDN besitzen muss. Für ihn reicht die Kenntnis der URL der von ihm angeforderten Datei. Die Serverauswahl geschieht implizit durch die Domainname-IP-Adresse-Auflösung.

Dispatcher basiert. Dieser Ansatz ist dem DNS basierten sehr ähnlich: Sämtliche Anfragen werden an einen Dispatcher geleitet, dem alle nötigen Informationen über die Kopieserver zur Verfügung stehen. Einem Client ist also nur die Adresse dieses Dispatchers bekannt. Nach außen sieht der Dispatcher für den Client wie der Originalserver aus. Der Dispatcher leitet die bei ihm ankommenden Anfragen an den für den anfragenden Client besten Server weiter. Hauptunterschied zum DNS basierten Ansatz ist, dass die Anfrageumleitung nicht nur auf IP-Ebene erfolgen muss, sondern auf URL-Ebene geschehen kann. Die Anfrageumleitung des Dispatchers kann unter anderem auf folgende Art und Weise geschehen:

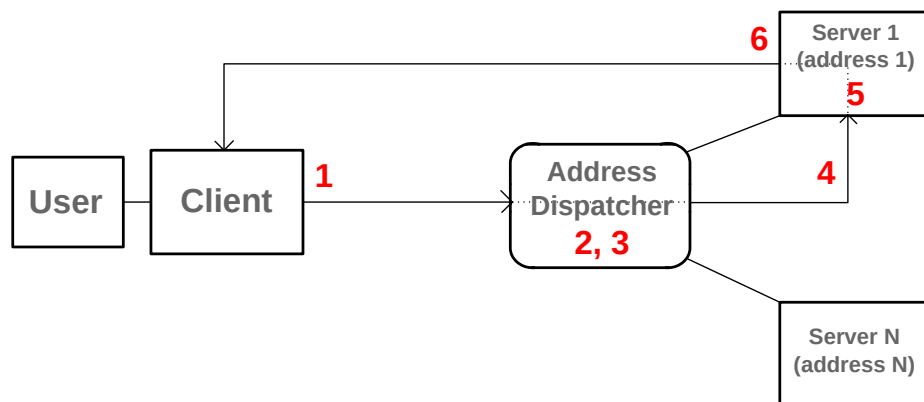


Abbildung 5: **Packet Single-Rewriting**. Schritt 1: Anfrage des Dokuments. Schritt 2: Auswahl eines Servers durch den Dispatcher (hier Server 1). Schritt 3: Umschreiben des TCP-Headers (Dispatcheradresse → Serveradresse). Schritt 4: Weiterleitung der Clientanfrage an Server 1. Schritt 5: Server 1 schreibt die Adresse des Dispatchers als Quelladresse der Antwort in den TCP-Header. Schritt 6: Server 1 sendet das angeforderte Dokument. Abbildung aus [4].

- **Packet Single-Rewriting.** Bei diesem Verfahren ersetzt der Dispatcher die Ziel-IP-Adresse im Transmission Control Protocol (kurz: TCP) Header des bei ihm ankommenden Pakets mit der IP-Adresse desjenigen Spiegelservers, an den das Paket weitergeleitet werden soll. Abbildung 5 aus [4] veranschaulicht dies. Eine derartige Funktionalität findet man auch in einfachen TCP-Routern.

Zu beachten ist, dass der Replica Server in der Antwort an den Client nicht seine eigene Adresse vermerkt sondern die des Dispatchers. Dies ist notwendig, da der Client die Antwort vom Dispatcher erwartet, da er an ihn auch die Anfrage gestellt hat.

Problematisch ist jedoch, dass die implizite Fehlerkorrektur für doppelt versandte und verloren gegangene Pakete, die TCP mit sich bringt, nicht mehr ohne weiteres funktioniert, da der Replica Server seine wahre IP-Adresse verbirgt.

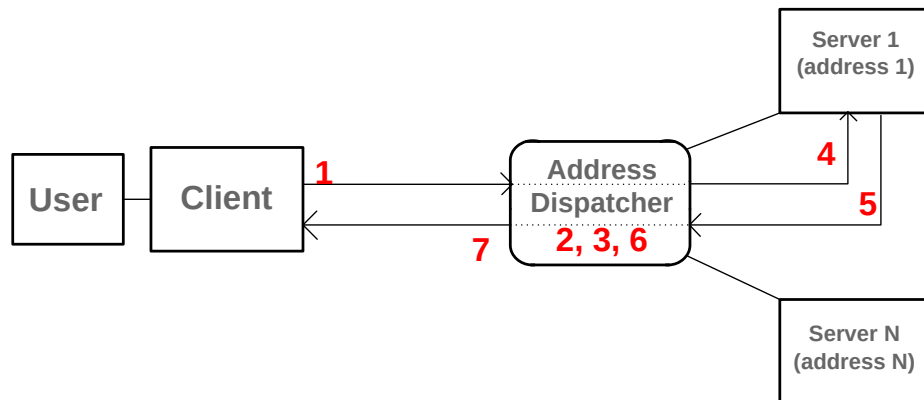


Abbildung 6: **Packet Double-Rewriting**. Schritt 1: Anfrage des Dokuments. Schritt 2: Auswahl eines Servers durch den Dispatcher (hier Server 1). Schritt 3: Umschreiben des TCP-Headers (Dispatcheradresse → Serveradresse). Schritt 4: Weiterleitung der Clientanfrage an Server 1. Schritt 5: Server 1 sendet jedes Antwortpaket an den Dispatcher. Schritt 6: Dispatcher schreibt den TCP-Header um (Serveradresse → Dispatcheradresse). Schritt 7: Der Dispatcher leitet das angeforderte Dokument weiter an den Client. Abbildung aus [4].

- **Packet Double-Rewriting**. Genau so wie beim Packet Single-Rewriting modifiziert der Dispatcher die Ziel-IP-Adresse. Darüber hinaus ersetzt er die Quell-IP-Adresse (also die IP-Adresse des anfragenden Clients) mit seiner eigenen. Dies hat zur Folge, dass alle Antworten der Replica Server zuerst über den Dispatcher zurück laufen, bevor sie an die Clients gesendet werden (siehe Abbildung 6 aus [4]).

Motivation für dieses Verfahren ist die Tatsache, dass nicht wie beim Packet Single-Rewriting der Spiegelservers als Absenderadresse die des Dispatchers eintragen muss. Logisch laufen ja wieder alle Antwortpakete über den Dispatcher, bevor sie an den anfragenden Client geleitet werden. Somit funktioniert auch wieder die Erkennung und Korrektur von Paketverlusten durch TCP.

Diese Auflistung der Methoden der Dispatcher basierten Anfrageumleitung ist keineswegs vollständig. Es sind diverse andere Techniken vorstellbar, wie sie in [4] vorgestellt werden.

Server basiert. Dieses Verfahren ist dem DNS und Dispatcher basierten Verfahren recht ähnlich, jedoch entlastet es den Flaschenhals des DNS Servers beziehungsweise des Dispatchers.

Beim Server basierten Ansatz handelt es sich um ein Zwei-Level-Verfahren: Alle Anfragen gehen zunächst wieder beim primären DNS Server des CDN oder einem Dispatcher ein, welcher die Anfragen mit den oben vorgestellten Verfahren an die zur Verfügung stehenden Spiegelservers weiterleitet. Im zweiten Schritt kann jeder einzelne ausgewählte Server die bei ihm eingehende Anfrage an einen

anderen Server weiterleiten, je nachdem ob dies sinnvoll ist oder nicht. Die Aufgabe der Lastverteilung im CDN liegt also nicht mehr allein beim DNS Server beziehungsweise beim Dispatcher.

Der Nachteil dieses Verfahrens besteht allerdings darin, dass jeder einzelne Spiegelserver Informationen über die anderen Spiegelserver benötigt, damit eine effektive Weiterleitung von statten gehen kann. Bei den anderen Ansätzen reicht es, wenn der DNS Server beziehungsweise der Dispatcher über diese Informationen verfügen.

6 Schluss

Diese Arbeit gibt einen grundlegenden Überblick über Content Delivery Networks und deren Funktionsweise. Sie zeigt Probleme klassischer Methoden der Dateiverteilung auf und wie diese durch Content Delivery Networks gelöst werden können. Anschließend wird erläutert, welche Algorithmen die in einem CDN nötige Funktionalität zur Verfügung stellen. Dabei handelt es sich zunächst um die für die Clients optimale Verteilung der Kopien auf die Spiegelserver und anschließend um die Auswahl des optimalen Spiegelserver für einen konkreten Client.

Es bleibt aber kritisch anzumerken, dass die meisten hier vorgestellten Algorithmen genaue Kenntnisse über das Verhalten der Clients und die Topologie des Netzwerks voraussetzen. Dazu gehören unter anderem qualitative Aussagen über die Verbindung der Clients zu den Spiegelservern, noch bevor die Clients auf diese zugreifen. In der Realität stehen solche Daten jedoch nicht initial zur Verfügung. Um im Voraus über derartige Informationen zu verfügen, muss auf Erfahrungswerte oder stochastische Simulationen zurück gegriffen werden.

Darüber hinaus stellen CDN in der obigen Form keine wirklich neue Technologie dar. Sie bauen noch immer auf zentralen Servern auf, auch wenn diese nun zahlreich und global verteilt sind. Jedoch kann eine genügend große Nutzerzahl diese Kopieserver immer noch überlassen. Technologien wie Peer-to-Peer-Netze, welche die Anzahl der Nutzer, die eine Ressource anbieten mit der Anzahl der Nutzer, die eine Ressource in Anspruch nehmen, skalieren, können für solche Szenarien besser geeignet sein.

Literatur

1. Novella Bartolini, Emiliano Casalicchio, Salvatore Tucci: „A Walk through Content Delivery Networks“, in Performance Tools and Applications to Networked Systems, volume 2965 of Lecture Notes in Computer Science, Springer, 2004.
2. Lili Qiu, Venkata Padmanabhan, Geoffrey Voelker: „On the Placement of Web Server Replicas“, in Infocom, pages 1587-1596, 2001.
3. Sugih Jamin, Cheng Jin, Anthony Kurc, Danny Raz, Yuval Shavitt: „Constrained Mirror Placement on the Internet“, in Infocom, pages 31-40, 2001.
4. Valeria Cardellini, Michele Colajanni, and Philip Yu: „Dynamic Load Balancing on Web-Server Systems“, in IEEE Internet Computing, 3(3):28-39, May-June 1999.
5. Mehmet Sayal, Yuri Breithart, Peter Scheuermann, Radek Vingralek: „Selection Algorithms for replicated Web Servers“, December 1998.