Bandbreitenreduktion durch (lokale) Caches

Armin Slopek
Hochschule Bonn-Rhein-Sieg
Fachbereich Informatik
Grantham-Allee 20
53757 Sankt Augustin
E-Mail: armin.slopek@smail.inf.h-brs.de

Thomas Thaeren
Hochschule Bonn-Rhein-Sieg
Fachbereich Informatik
Grantham-Allee 20
53757 Sankt Augustin
E-Mail: thomas.thaeren@smail.inf.h-brs.de

***Abstract*—Die im Internet angebotenen, stetig weiterentwickelten Inhalte und Services werden immer umfangreicher. Demgegenüber stehen beim Transport der Daten vom Service Provider zum Endanwender jedoch (insbesondere in abgelegenen Gebieten) nicht immer die hierfür notwendigen Hochgeschwindigkeits-Datennetze zur Verfügung. Neben dem Netzausbau zur Steigerung der Übertragungsraten ist es daher von elementarer Bedeutung, Ansätze zur Verringerung des Datenaufkommens zu implementieren: Caching. In dieser Ausarbeitung werden hierzu die theoretischen Hintergründe erörtert, eine zu diesem Thema bereits durchgeführte Studie näher beleuchtet sowie mit den Ergebnissen eines eigenen Experiments verglichen. Im Rahmen des Experiments konnte durch den Einsatz von Caching eine Reduktion der Bandbreitennutzung (Effektivität) von sieben Prozent sowie Kosten im Verhältnis von 2:1 bei der Cacheauslastung gegenüber der Effektivität festgestellt werden. Bildinhalte zeigten sich hierbei als besonders lohnenswert, gecacht zu werden. Anfragen verschlüsselter Inhalte konnten hingegen zu keinem Zeitpunkt aus dem Cache beantwortet werden.**

Schlagwörter: Caching, Reduktion Bandbreitennutzung, Caching verschlüsselter Inhalte

I. Einleitung und Motivation

Der Siegeszug moderner Informations- und Telekommunikationssysteme (ITK-Systeme) ist unaufhaltsam. Die Digitalisierung stößt mittlerweile in fast alle Bereiche des täglichen Lebens vor, wodurch sie die Kommunikation und Interaktion in der heutigen Zeit maßgeblich prägt – und somit einen wesentlichen Grundpfeiler unserer Gesellschaft darstellt. In den letzten Jahren ist insbesondere im mobilen Datensektor ein exponentieller Anstieg zu verzeichnen; so prophezeit etwa Cisco in 2014 [1]: „In vier Jahren wird es in Deutschland doppelt so viele Mobilgeräte geben wie Einwohner". Doch nicht nur Mobiltelefone, sondern auch die Verwendung von Big Data und die Entwicklung des Internet of Things (IoT) werden immer intensiver vorangetrieben [2]. Diese neuen Technologien sowie deren erhöhte Verbreitung stellen die Betreiber von ITK-Systemen vor große Herausforderungen, wie etwa durch das enorm steigende Datenaufkommen bei limitierter Bandbreite.

Dem erhöhten Datenaufkommen und der elementaren Bedeutung der ITK-Systeme Rechnung tragend, ist es folglich notwendig, dieser Entwicklung mit dem Ausbau und der Modernisierung eben jener Systeme zu begegnen. Ein weiterer – bereits etablierter – Ansatz besteht darin, durch Caching-Strategien den Bandbreitenbedarf zu entlasten. Hierbei werden bestimmte Datenobjekte zwischengespeichert, um neben der Bandbreite auch Anfragezeiten drastisch reduzieren zu können; dies kann beispielsweise auch in ländlichen Gebieten sinnvoll sein, in denen große Entfernungen durch kostenintensive Satelliten-Links bewältigt werden müssen.

Obwohl das Caching bereits seit geraumer Zeit praktiziert wird, haben sich in den letzten Jahren aufgrund neuer Anforderungen alternative Caching-Strategien entwickelt, weshalb es geboten erscheint, den Stand der Technik herauszuarbeiten und die Vielzahl an Literatur kritisch zu würdigen. Daher setzt sich die folgende Arbeit mit dem Themenfeld Caching auseinander.

Zu Beginn wird hierbei eine eigene einheitliche Terminologie geschaffen, die es ermöglicht, verschiedene Technologien auf einer abstrakten Ebene zu betrachten, um so ungeachtet der eingesetzten Technologie eine objektive Vergleichbarkeit zu ermöglichen und das Verständnis zu festigen. Im Anschluss wird der Stand der Technik herausgearbeitet und Ansätze kritisch hinterfragt. Hierbei soll der Fokus dieser Arbeit auf dem Caching der Ebenen bis zum Network Access Node (siehe Abbildung 1) liegen. Fragestellungen, die behandelt werden sind:

- Welche Datenobjekte eignen sich zum Cachen?
- Wie viel Gewinn ist zu erwarten?
- Lohnt es sich im Zeitalter der Kryptographie überhaupt noch, Caching zu verwenden?

Um diese theoretischen Ausarbeitungen zu unterstützen, sollen bereits bekannte Ansätze für das Caching in einem Hardwaremodul implementiert werden, um somit die Vorzüge des Cachings quantifizieren und das thematische Verständnis festigen zu können.

II. Terminologie

In der Literatur finden sich im Bereich des Themenfeldes Caching diverse Begrifflichkeiten für analoge Sachverhalte. Dies liegt vornehmlich darin begründet, dass die meisten Publikationen auf eine bestimmte Technologie abzielen – wie etwa 3G oder 4G, aber auch auf kabelgebundene Ansätze. Des Weiteren lassen sich Unterschiede im jeweiligen Detailgrad der betrachteten Ebenen ausmachen. Diese unterschiedliche

Terminologie erschwert das Verständnis von Caching erheblich. Dabei ist eine terminologische Differenzierung aufgrund verschiedener Technologien nicht immer notwendig, da alle Ansätze auf vergleichbaren Grundstrukturen basieren. Die nachfolgende Graphik schlägt daher eine Brücke und baut eine eigene einheitliche Terminologie für die unterschiedlichen Caching-Ebenen auf, die im Verlaufe dieser Seminararbeit ihre Anwendung findet:

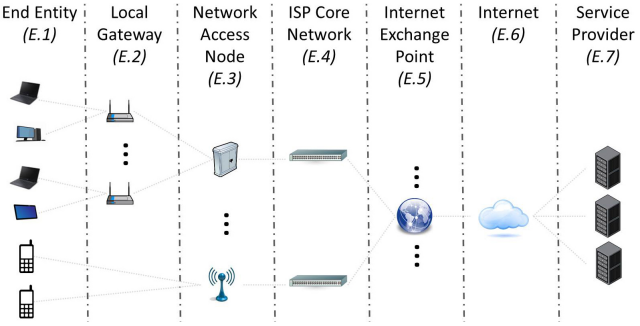

Abbildung 1: Darstellung der Netzwerkebenen (Eigene Abb.)

Wie die Abbildung 1 darstellt, befinden sich von links nach rechts auf der ersten Ebene die Endgeräte – die sogenannten End Entities. Diese können in ihrer Ausprägung sowohl mobile Geräte wie Smartphones, Tablets oder Laptops, als auch stationäre wie PC, Smart TV oder Blu-ray-Player darstellen.

Auf der zweiten Ebene sind die lokalen Gateways (local Gateway) angesiedelt. Diese obliegen der direkten räumlichen Kontrolle der Endverbraucher, i.d.R. sind dies Router oder Modems; mobile Geräte können mit diesen ebenso kommunizieren wie stationäre. Je nach Internet Service Provider (ISP) kann es jedoch vorkommen, dass dieser einen gesonderten Zugang, z.B. über bereitgestellte Netzwerkhardware, erhalten kann.

Die Local Gateways sind in Ebene 3 direkt mit dem Network Access Node (NAN) verbunden. Dieser wird im mobilen Bereich auch mit Radio Access Network (RAN) [3, 4] bezeichnet. Weitere mögliche Begriffe sind Small Base Station (SBS) [5, 6, 7] oder im konkreten Fall von 3G und 4G NodeB (NB) bzw. evolved NodeB (eNB) [6, 4, 8, 9]. In kabelgebundenen Infrastrukturen – wie etwa bei DSL – wird diese Instanz auch als Digital Subscriber Line Access Multiplier (DSLAM) bezeichnet.

Die verschiedenen NANs sind wiederum über einen Backhaul-Link [5, 6, 3, 4, 8, 7] mit dem ISP Core Network verbunden (Ebene 4). In dem konkreten Fall von mobilen Netzwerken handelt es sich dabei um den Evolved Packet Core (EPC) [4]. Das ISP Core Network ist in Ebene 5 mit den Internet Exchange Points (IXP) verbunden, welche die Endkunden letztlich in der siebten Ebene mit den Service Providern (SP) zusammenbringen.

III. Stand der Technik

Dieses Kapitel bietet einen Überblick über den aktuellen Stand der Technik im Bereich Caching. Dieser beginnt mit einer Vorstellung der allgemeinen Funktionsweise eines Cachingvorganges sowie der Herausarbeitung der Vor- und Nachteile. Anschließend werden die verschiedenen Cache-Positionierungsmöglichkeiten unter Berücksichtigung der jeweiligen Effekte aufgezeigt und diskutiert. Es folgt eine Definition, bei welchen Datenobjekten es sinnvoll erscheint zu cachen. Darauf aufbauend werden im Anschluss Caching Architekturen betrachtet, die eine Zusammenarbeit verschiedener Caches erlauben sowie drei grundlegende Strategien vorgestellt, wie Caching von der technischen Seite her umgesetzt werden kann: Die Erfassungs-Strategie, die Daten-Organisations-Strategie und zuletzt die Cache Ersetzungs-Strategie.

A. Caching im Allgemeinen

Cachingtechnologien ermöglichen durch temporäres Vorhalten strategischer Objekte (siehe Kapitel Strategische Objekte) eine Eliminierung redundanter Datenübertragung im Netz [10]. Nachfolgende Abbildung illustriert die generische Arbeitsweise eines Caches, der an einem Knotenpunkt betrieben wird. Es folgt eine anschließende Erläuterung der individuellen Schritte; im vorliegenden Beispiel wird davon ausgegangen, dass ein angefragtes Datenobjekt nicht im Cache vorhanden ist:

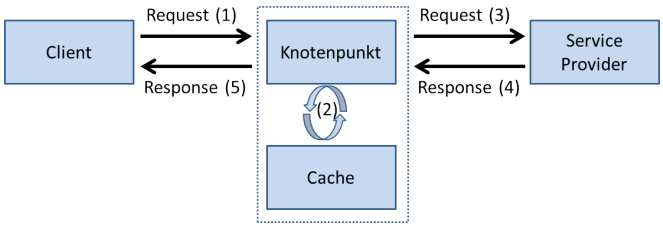

Abbildung 2: Generische Arbeitsweise eines Caches (Eigene Abb.)

1) Der Client initiiert einen Request auf ein Datenobjekt, welches durch einen Service Provider zur Verfügung gestellt wird.
2) Der Knotenpunkt unterbricht den Request und konsultiert den Cache.
3) Das angefragte Datenobjekt ist nicht im Cache vorhanden. Der Request wird an den Service Provider weitergeleitet.
4) Der Service Provider liefert das Datenobjekt dem Knotenpunkt aus; dieses wird im Cache für zukünftige Anfragen persistiert.
5) Das angefragte Datenobjekt wird nun an den Client zurückgesendet und stünde bei einer nachfolgenden Anfrage bereits im Cache zur Verfügung. Dies hätte zur Folge, dass Schritte 3 und 4 entfallen würden.

Aus dem oben beschriebenen Szenario lassen sich nun etwaige allgemeine Vor- und Nachteile des Einsatzes von Caching ableiten; eine weitergehende Beschäftigung mit dieser Thematik ist unter [11] zu finden. Die Anwendung von Caching hat in erster Linie eine Reduzierung der Netz- und

Serverauslastung zur Folge, da häufig getätigte Anfragen nicht zwangsläufig über den Knotenpunkt hinaus gesendet werden müssen. Gleichzeitig kann für den Fall, dass Datenobjekte im Cache gefunden werden, auch die Netzlatenz deutlich reduziert werden – das Wegfallen der Übertragungsstrecke zwischen Knotenpunkt und Service Provider hätte so etwa eine kürzere Wartezeit für den Client zur Folge. Ein weiterer wesentlicher Vorteil von Caching besteht zugleich darin, dass die Verfügbarkeit von Datenobjekten steigt: Ist der Zugriff auf den Service Provider temporär unterbunden, so könnten gecachte Datenobjekte jedoch weiter ausgeliefert werden.

Andererseits birgt der Einsatz von Caching auch Risiken. So können Konsistenzprobleme im Cache auftreten, wenn sich Datenobjekte vom Service Provider geändert haben und dies vom Cache unbemerkt bleibt. Daher müssen entsprechende Maßnahmen getroffen werden, um Inkonsistenzen zu vermeiden und die Aktualität der Daten zu gewährleisten. Ferner ergeben sich aus dem Einsatz von Caches weitere Angriffsvektoren für Angreifer; beispielsweise das Einfügen gefälschter (Bank-)Seiten, welche anstelle der echten Seiten an die Nutzer ausgeliefert werden. Dieser Punkt findet jedoch überraschenderweise in der gegenwärtigen Literatur nur wenig Berücksichtigung, während die Anzahl der Bedrohung stetig steigt. Neben Konsistenzproblemen kann sich bei einer schlechten Hit Ratio[1] die Verbesserung der Netzlatenz ins Gegenteilige umkehren: Unnötiges Nachschlagen im Cache bei Objekten, die zuvor nicht gespeichert wurden, hat zur Folge, dass diese ohne den Einsatz von Caching ggf. schneller hätten geladen werden können.

Zusammenfassend lässt sich durch Caching eine höhere Geschwindigkeit bei der Diensterbringung ausmachen, die zugleich geringere Pfadkosten nach sich zieht, wobei jedoch Probleme mit der Datenkonsistenz entstehen können.

Die genannten positiven Faktoren stehen hierbei jedoch in unmittelbarer Abhängigkeit einer strategisch guten Platzierung des Caches. Wie in Kapitel 2 Terminologie erläutert, umfasst das Internet bzw. der Pfad vom Endgerät eines Anwenders (End Entity) bis zum Server eines Service Providers mehrere Knoten, welche sich hierarchisch in Ebenen einsortieren lassen. Prinzipiell kann an allen diesen Knotenpunkten ein Cache installiert werden. Es liegt jedoch im besonderen Interesse des ISPs zur Kosteneinsparung und Steigerung der Durchsatzraten Caching einzusetzen, weshalb das Caching auf vom ISP kontrollierten oder kontrollierbaren Ebenen von Bedeutung ist. Bei der Wahl der optimalen Position kommt es insbesondere auf den gewünschten Effekt an, beispielsweise Kostenersparnisse und/oder Performance-Gewinne.

[1]Anteil der Anfragen die vom Cache bedient werden konnten: Hit Ratio=number of hits / (number of hits + number of miss)

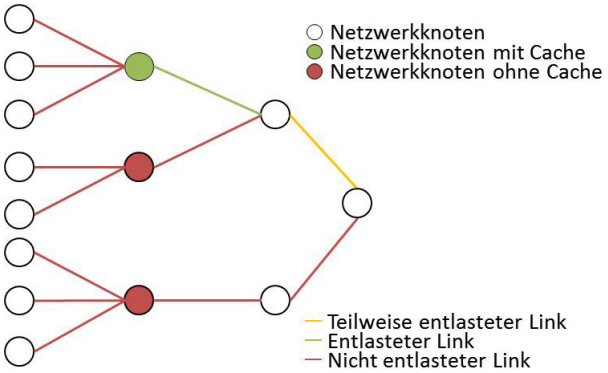

Abbildung 3: Generische Netzwerk-Hierarchie mit Caching auf einer Ebene (Eigene Abb.)

Aus der Abbildung 3 wird ersichtlich, dass ein Cache insbesondere den direkt hinter ihm liegenden Link entlastet (Kostenersparnis). In der Graphik dargestellt ist ein Netzwerk bestehend aus mehreren Knoten. Betrachtet wird das Caching auf einer bestimmten Ebene. Der Cache kann hauptsächlich die direkt hinter ihm liegende Leitung (in grün dargestellt) zum nächsten Knoten entlasten; wenn auch die darauf folgenden Links ebenfalls entlastet werden, so ist dies relativ gesehen nicht ausschlaggebend, da hier auch Anfragen von weiteren Knoten eintreffen. Wird folglich der Cache in der Hierarchie weiter oben platziert, ergeben sich mehr Vorteile für den Netzbetreiber: Soll beispielsweise eine ländliche Region mit Internet über einen einzigen Link versorgt werden – z.B. per Satellit – ergeben sich für den Netzbetreiber Kostenersparnisse bei der Nutzung dieses Links.

Je näher der Cache hingegen an der End-Entity bzw. dem Nutzer liegt, desto mehr Performance-Gewinn kann dieser verzeichnen. Wird beispielsweise auf Heimnetzwerk-Ebene gecacht – d.h. in obiger Graphik sei der grün markierte Netzwerkknoten das local Gateway – so können Nutzer mit schwacher Internetanbindung davon profitieren.

B. Strategische Objekte

1) Cachbare Daten: Nachdem eine strategisch günstige Position eines Caches ausgemacht und diskutiert wurde, wird an dieser Stelle betrachtet, bei welchen Datenobjekten es überhaupt sinnvoll erscheint Caching zu verwenden. Aufgrund der großen Vielfalt möglicher Objekte wird eine negierte Definition gegeben. Hierbei wird zunächst davon ausgegangen, dass alle Objekte lohnenswert sind gecacht zu werden – von dieser Menge wird dann explizit eine bestimmte Gruppe ausgeschlossen. Bevor diese definitiv dargelegt wird, erfolgt zum besseren Verständnis ein kurzer Überblick über den wesentlichen Aufbau einer HTTP-Nachricht [12, 13].
Eine HTTP-Nachricht beginnt mit einer *Start-Line*, in der grundlegende Informationen übertragen werden. Da es sich beim HTTP um ein Request-/Response Protokoll handelt, wird in der Start-Line in Abhängigkeit des jeweilen Nachrichtentyps entweder beim HTTP-Request die HTTP-Methode (z. B. GET, POST, DELETE) oder im Falle des HTTP-Response der Status-Code mitgeteilt. Jede HTTP-Response

lässt sich hierbei einer der folgenden fünf Klassen zuordnen: Informational=1xx, Success=2xx, Redirect=3xx, Client-Error=4xx, Server-Error=5xx. Im Anschluss wird in beiden Fällen die jeweilige HTTP-Versionskennung mitgeliefert. Der nachfolgende Header-Block besteht aus einer bestimmten Menge an Header-Zeilen. Während die Header-Zeilen des Request-Headers dem HTTP-Server Auskunft darüber geben, wie eine akzeptierte Nachricht beschaffen sein sollte, dienen die Header-Zeilen des Response-Headers zur Ergänzung der Start-Line; es werden Informationen über den HTTP-Server und den Zugriff auf das angefragte Datenobjekt platziert. Getrennt durch eine Leerzeile werden im optionalen Body die Nutzdaten übertragen.

Ramanan et al. [8] bieten eine Übersicht an Eigenschaften von HTTP-Response-Paketen, deren Inhalte nicht gecacht werden (können); HTTP-Requests können im allgemeinen nicht gecacht werden. Diese Eigenschaften werden in folgender Tabelle auszugsweise betrachtet:

Eigenschaft	Bedeutung
Set-cookie non-NULL im HTTP-Header	Personalisierte Daten durch gesetzten Cookie
Content-Length = 0 im HTTP-Header	Länge des Body ist 0 (keine Nutzdaten)
Cache control = “private“ / “no-store“ / “no-cache“ im HTTP-Header	Explizite Aufforderung die Pakete nicht zu cachen
HTTP-Status-Code nicht {200, 203, 206, 300, 301, 410}	Enthält der HTTP-Header keinen dieser Status-Codes, so bedeutet dies, dass ein Paket nicht gecacht werden sollte.

Tabelle I: Ausschlusskriterien für nicht-cachbare HTTP-Objekte (nach Ramanan et al. [8])

Letztere Einschränkung bedeutet insbesondere, dass verschlüsselte – also mittels HTTPS übertragene – Daten nicht zu den cachbaren Daten gehören. Dies liegt in der Tatsache begründet, dass bei HTTPS (SSL/TLS) Paketen die Status-Codes aufgrund der Ende-zu-Ende-Verschlüsselung nicht sichtbar sind (siehe auch Kapitel Caching im Kontext der Kryptographie). Dies ist technisch bedingt durch die Kapselung der HTTP-Pakete inkl. Header in den TLS-Paketen, sodass ohne Kenntnis des jeweiligen Session-Keys die HTTP-Status-Codes, aber auch die weiteren Eigenschaften des Headers, nicht eingesehen werden können. Da die Anfragen ebenfalls verschlüsselt sind, wäre es ohnehin nicht möglich, nach passenden Objekten im Cache zu suchen.

Die Einstellung “private“ im Bereich “Cache control“ bedeutet, dass die übertragenen Daten nur für einen bestimmten Nutzer gecacht sind, z.B. bei der Übertragung personenbezogener Daten. Mit dieser Option soll verhindert werden, dass die übertragenen Datenpakete in zwischengeschalteten Caches – also Caches, welche sich an Netzwerkknoten auf dem Pfad zwischen Nutzer und Server befinden – abgelegt werden. Im lokalen Cache des Nutzers, also in seinem Browser, können diese Daten jedoch problemlos vorgehalten werden, da sie hier im Klartext vorliegen. Da wir uns im weiteren Verlauf jedoch mit zwischengeschalteten Caches auseinandersetzen wollen, werden wir Objekte, deren Cache control auf private gesetzt ist, als nicht cachbar ansehen.

Zusätzlich zu dieser Option können die Optionen “no-cache“ und “no-store“ gesetzt werden, jedoch nicht zu gleicher Zeit. Mit “no-cache“ versehene Datenpakete können zwar prinzipiell gecacht werden, jedoch wird vor der Auslieferung des gecachten Inhalts eine Validierung der Aktualität erzwungen. Da sich diese Option besonders für sich häufig ändernde Inhalte eignet und i.d.R. auch nur für solche gesetzt wird, betrachten wir diese ebenfalls als nicht cachbar. Im Zweifelsfall kann hieraus eine erhöhte Bandbreitennutzung resultieren, wenn sich nach der Validierung herausstellt, dass sich die zwischengespeicherten Daten geändert haben. Im Fall von “no-store“ wird in jedem Fall die Anfrage an den entsprechenden Server weitergeleitet und die Antwort vom Server an den Client weitergereicht; demzufolge wäre es Ressourcenverschwendung diese Antworten zu cachen [14].

2) Inhalts-Typen: Der HTTP-Datenverkehr besteht im Wesentlichen aus fünf Kategorien übertragener Daten: Bild, Text, Applikation (Sonstige), Audio, Video.

In einem Feldversuch von Ramanan et al.[8], welcher 2012 in einer nordöstlichen Metropolregion der Vereinigten Staaten durchgeführt wurde, zeigten sich die folgenden Eigenschaften hinsichtlich der Betrachtung, ob es lohnenswert ist einen bestimmten Inhalts-Typ zu cachen:
Bilddateien ließen sich aufgrund wiederkehrender Anfragen und ihrer hohen Lebensdauer effektiv cachen. Dies können beispielsweise Logos von populären Webseiten sein.
Textinhalte waren zwar ebenso wie Bilddateien häufig angefragt, jedoch war der hieraus gezogene Nutzen geringer, da Texte vergleichsweise geringe Ansprüche an die Bandbreite und somit die Übertragungskosten stellen.
Da, ebenso wie bei Text-Inhalten, hauptsächlich kleinere Objekte, nicht aber Große, mehrfach angefragt wurden, ergab sich für das Cachen der Applikations-Daten ebenfalls keine Entlastung des Backhaul-Links. Wie sich im zu Ende beschriebenen Experiment noch herausstellt, bestehen diese Applikationsdaten hauptsächlich aus JSON, JavaScript und XML-Dokumenten, verhalten sich also von Hause aus ähnlich wie Text-Inhalte.
Die Effektivität des Cachens von Audioinhalten konnte durch einen marginalen Anteil an Requests und einen nicht messbaren Anteil am Datenvolumen ebenfalls als schlecht bewertet werden. Der Anteil an Requests wurde mit einem Prozent beziffert, während der gemessene Anteil am Datenvolumen sogar unter einem Prozent lag.
Videoinhalte hatten indessen einen besonders hohen Anteil am abgefragten Datenvolumen. Diese zu cachen kann somit die Netzwerkauslastung in besonderem Maße verringern, stellt jedoch eine enorme Belastung des Speichervolumens des Caches dar.

Insgesamt konnte in dem Feldversuch eine Verringerung des Datenverkehrs über einen Backhaul-Link von 810 GB bei einem Gesamtaufkommen von 12,2 TB (von welchen neun TB bzw. 73,9% die Definiton von Cachebarkeit erfüllten) gemes-

sen werden, was einer Reduktion der Bandbreitennutzung (Effektivität) von 6,63% entspricht. Über den Versuchszeitraum hinweg wurden 42,6 Millionen Requests mit einem Gesamt-Datenvolumen von 12,2 TB aufgezeichnet, von welchen 13 Millionen Requests bzw. neun TB als cachbar identifiziert wurden.

Um bestimmen zu können, welche Inhalts-Typen es lohnt in welchem Maße zu cachen, sollten somit neben der Hit Ratio ebenfalls die Kosten für die Übertragung über den Backhaul-Link und die Kosten für das Speichern im Cache betrachtet werden.

3) Inhalts-Anbieter: Der aus der Speicherung von Inhalten gewonnene Nutzen ist nicht nur abhängig vom jeweiligen Inhalts-Typ, sondern in besonderem Maße ebenfalls vom Bereitsteller des Inhalts (Service Provider). D.h. mit steigender Popularität des Service Providers ist es als sinnvoll zu erachten, den angeforderten Daten eine höhere Priorität im Cache zuzuordnen, da diese Daten potentiell öfter abgerufen werden. So kann das Cachen von YouTube-Inhalten zur Entlastung des Backhaul-Links beisteuern, bewirkt jedoch einen immensen Overhead im Speicher des Caches. Das Speichern von News-Webseiten ist eine effektive Möglichkeit, den Netzwerkverkehr zu entlasten, bringt jedoch auch Revalidierungs-Overhead mit sich, da diese Inhalte öfters von Änderungen betroffen sein können [8]. Auch das Cachen bestimmter Inhalte Im Bereich Social Media, insbesondere die Inhalte einflussreicher User, kann aufgrund der hohen Aufruf-Rate den Link effektiv entlasten [8, 5].

C. Cache Architekturen

Da die Vorteile einzelner Caches stark limitiert sind, setzen heutige Systeme auf Caches, die untereinander zusammenarbeiten. Hierbei existieren zwei prominente Ansätze [15, 16, 17]: hierarchisches- und verteiltes Caching.

Beim hierarchischen Caching folgt der Aufbau einer Baumstruktur. Anfragen die zu einem Cache-Miss geführt haben, werden an den übergeordneten Knoten weitergeleitet. Dies erfolgt solange bis das gewünschte Datenobjekt gefunden wurde – spätestens bei dem jeweiligen Service Provider. Die Antwort wird entsprechend der Aufrufhierarchie zurückgereicht. Hierbei verbleibt auf jeder Ebene jeweils eine Kopie des zuvor angefragten Datenobjektes. Dies hat den Vorteil, dass niedrig stehende Caches von einer höheren Bandbreite der darüberliegenden Caches profitieren. Die Nachteile, die sich aus einer solchen Struktur ergeben, sind, dass jede Ebene zusätzliche Verarbeitungszeit hinzufügt und redundante Datenobjekte auf verschiedenen Ebenen vorgehalten werden. Zudem bilden Caches in höherliegenden Ebenen einen Flaschenhals, der zu Engpässen führen kann.

Im verteilten Caching hingegen kooperieren alle Caches gleichberechtigt; bei einem Cache-Miss werden angefragte Datenobjekte von benachbarten Caches geholt. Dies hat wiederum einen bedeutenden Kommunikationsoverhead zur Folge: die verschiedenen Caches müssen entweder bereits wissen, welcher Cache welche Objekte bereitstellt oder aber diese Informationen möglichst effizient bestimmen können. In [15] werden einige solche Mechanismen, wie etwa das Inter Cache Protocol (ICP), oder hashbasierte Verfahren vorgestellt. Vorteile, die sich aus diesem Ansatz ergeben, bestehen aus einer besseren Lastenverteilung und einem daraus resultierenden geringeren Speicherbedarf in den einzelnen Caches, da eine Vielzahl an Duplikaten vermieden werden kann. Dagegen muss man mit einem größeren administrativen Aufwand rechnen sowie einer zusätzlichen Belastung des Netzwerkes, die der Inter-Cache-Kommunikation geschuldet ist. Um die Vorteile beider Ansätze nutzen zu können, haben sich hybride Ansätze entwickelt, die im Grunde genommen auf einem hierarchischen Modell basieren, jedoch mit einer ausgewählten Anzahl an kooperierenden Caches pro Ebene agieren.

D. Cache Erfassungs-Strategien

Das Caching selbst kann unabhängig von der Cache Architektur auf mehrere Arten erfolgen. Hier ist zunächst das herkömmliche *passive Caching* (auch als *reaktives Caching* bezeichnet) zu nennen. Dieser Ansatz verfolgt das Ziel, bereits angefragte und geladene Inhalte zu cachen, um bei nachfolgenden Anfragen den Backhaul-Link zu entlasten und gleichzeitig Anfragezeiten zu reduzieren. Neue Ansätze wie das in [7] vorgestellte *proatkive Caching*, versuchen bereits im Voraus mögliche relevante Daten zu strategisch günstigen Zeitpunkten vorzuladen und stellen damit im Vergleich zum *passiven Caching* einen wesentlich mächtigeren Ansatz dar. Durch das Erstellen von Nutzerprofilen oder den Einsatz von Machine-Learning-Tools werden eben jene Daten vorhergesagt, welche die Nutzer besonders interessieren, bevor diese tatsächlich angefragt werden. Im Rahmen von sozialen Online-Netzwerken ist es ebenso möglich, die Beziehungen der Nutzer untereinander sowie angegebene Interessen auszunutzen, um diese bei der Entscheidungsfindung zu berücksichtigen. Das Anlegen von Nutzerprofilen und Vorhersagen des Nutzungsverhaltens wird bereits in vielen Kontexten verwendet. Dabei existieren zuweilen jedoch eher clientseitige Lösungen, wie etwa bei dem Browser Google Chrome. Dort werden, wenn ein Nutzer auf einer Seite surft, basierend auf dem bisherigen Nutzungsverhalten Links auf dieser Seite bereits im Hintergrund vorgeladen, sodass sie beim späteren Aufruf verfügbar sind. Die stetig steigende Anzahl mobiler Endgeräte hat zudem zur Folge, dass diese zukünftig immer mehr im Prozess des Cachings mit einbezogen werden sollen und Datenobjekte mittels Device-to-Device (D2D) Kommunikation zwischen den Endgeräten ausgetauscht werden könnten [5].

E. Daten-Organisations-Strategien im Cache

Chen et al. unterscheiden in [4] im Wesentlichen zwei verschiedene Arten, wie die Datenobjekte im Cache organisiert werden können: Web Caching und Redundancy Elimination – Web Caching wird im Folgenden hinsichtlich zweier unterschiedlicher Ausprägungen vorgestellt. Für einen tieferen Einblick sowie eine Analyse der Effizienz der hier vorgestellten Verfahren sei auf Jeong et al. [18] verwiesen.

1) URL-Based Web-Caching: Da neben dem normalen Web-Browsing auch weitere internetfähige Anwendungen auf HTTP basieren, müssen entsprechend viele HTTP-Inhalte gecacht werden. Hierfür existiert ein Ansatz, der die URL als Referenzwert für ein Datenobjekt interpretiert; ein entsprechender Zugriffszähler erfasst zugleich Statistiken über die Anzahl der Anfragen.

Diese Art des Speicherns ist jedoch mit Risiken behaftet. So kann das sogenannte *Aliasing* von Inhalten, d.h. das Bereitstellen desselben Inhalts unter mehreren URLs, nicht erkannt werden. Als Konsequenz folgt, dass einerseits weniger Bandbreite eingespart wird, da derselbe Inhalt öfters abgefragt werden würde und zugleich eine Zunahme des Speicherbedarfes zu verzeichnen wäre, da derselbe Inhalt mehrfach gespeichert werden würde.

Auch das Caching temporärer Inhalte bereitet Probleme, da diese zwar den Cache belasten können, sich jedoch keine Bandbreitenreduktion aus ihrer Speicherung ergibt. Insbesondere bei sich häufig ändernden Inhalten muss der Cache die Konsistenz seiner Inhalte überprüfen und diese aktualisieren.

2) Prefix-Based Web-Caching: Das Prefix-Based Web-Caching ist eine weiterentwickelte Version des zuvor beschrienen URL-Based Web-Caching. Es erkennt doppelte Inhalte (neben der URL oder der Betrachtung der ersten N Zeichen einer URL) anhand eines *Prefix Key* – einem Hashwert über die ersten M Zeichen des Datenobjektes und des content-length Feldes im HTTP-Header. Hierbei ist die Wahl der Zahl M zu beachten: Bei geringem M können viele False Positives entstehen, d.h. Inhalte werden als gleich angesehen, obwohl sie ungleich sind; ist M jedoch zu groß dimensioniert, wird die Rechenkapazität des Caches deutlich mehr belastet.

3) Redundancy-Elimination: Anders als andere Caching-Strategien ist Redundancy Elimination (RE) nicht protokoll- oder applikationsabhängig, da ausschließlich Byte-Strings – also die „Rohdaten“ – betrachtet werden. Zur Umsetzung wird nicht an einer Stelle im Netzwerk (typischerweise ein Netzwerkknoten), sondern an zwei Stellen im Netz jeweils ein Cache installiert; diese Caches bilden logisch gesehen eine Einheit. Der Cache, welcher in der Netzhierarchie weiter oben steht, teilt den eingehenden Bytestream auf. Von den hieraus entsehenden *Chunks* wird der Hashwert berechnet und zusammen mit der Länge des originalen Chunks an den zweiten Cache gesendet. Dieser überprüft, ob er bereits einen zu Hashwert und Länge passenden Bytestream gespeichert hat und liefert ihn im Erfolgsfall aus. Hierdurch werden ausschließlich die Links zwischen den beiden Cache-Plattformen entlastet. Die Entlastung ist jedoch über die gesamte Länge aller Links hinweg gleichbleibend effektiv.

F. Cache Ersetzungs-Strategien

Bedingt durch den limitierten Speicherplatz eines Caches sind effiziente Ersetzungsstrategien für die Ersetzung von alten durch neue Inhalte erforderlich.

Traditionelle Verfahren, die ihren Ursprung in der Speicherverwaltung besitzen, sind etwa:

Least Recently Used (LRU): Die am längsten ungenutzten Einträge werden ersetzt.

Least Frequently Used (LFU): Die am wenigsten genutzten bzw. abgefragten Inhalte werden ersetzt.

First in First Out (FIFO): Die zuerst in den Cache aufgenommenen Inhalte werden auch zuerst wieder ersetzt.

Diese vorgestellten Verfahren werden in allen gängigen Proxy-Lösungen wie etwa Squid [19] umgesetzt.

Alternative Ansätze basieren auf der Ordnung objektspezifischer Eigenschaften, wie die Größe des Objektes oder die Zeit, die benötigt wird, um dieses neu zu beschaffen.

IV. Caching im Kontext der Kryptographie

Wie bereits illustriert, können HTTP-Status-Codes nicht ohne Weiteres aus Ende-zu-Ende-verschlüsselten Paketen gelesen werden; diese Eigenschaft trifft auch auf den Rest des HTTP-Headers sowie die eigentlichen Nutzdaten zu.

Wie in Abbildung 4 zu sehen, kann ein Betrachter zwar erkennen, dass ein mittels SSL verschlüsseltes Datenpaket HTTP-Informationen beinhaltet, es können jedoch weder der Header noch die Nutzdaten eingesehen werden; diese sind unter „Encrypted Application Data“ enthalten. Die HTTP-Daten sind vollständig in die TLS-Verschlüsselung eingebettet. Hierin liegt die Tatsache begründet, dass durch die in Kapitel Cachbare Daten an cachbare HTTP-Antworten gestellten Anforderungen solche Antworten ausgeschlossen sind, welche verschlüsselt übertragen wurden.

Da mit jeder Session ein neuer Session-Key zwischen Client (User) und Server (Service Provider) vereinbart wird, ist es zudem nicht nutzbringend, die verschlüsselten Antworten im Cache zu hinterlegen. Denn selbst bei gleichen Nutzinhalten sähe das erwartete Paket durch den neuen Verschlüsselungs-Key anders aus als ein bereits übertragenes und gecachtes Paket. Solange dem Cache der Schlüssel, durch welchen die Sitzung verschlüsselt wird, nicht bekannt ist, kann dieser den Inhalt nicht einsehen und daher nicht zwischenspeichern.

Es ist jedoch nicht gänzlich unmöglich, verschlüsselte Inhalte zu cachen. Verbindet sich ein Client nicht „direkt“, d.h. ohne festgelegte Knotenpunkte, sondern über einen Proxy-Server mit dem Internet, so kann der Proxy-Server die verschlüsselt übertragenen Daten mitlesen.

In diesem Fall stellt der Client nicht selbst die Anfrage an den Ziel-Server, sondern er beauftragt den Proxy damit, die Anfrage für ihn zu stellen und die Antwort weiterzuleiten. D.h. wenn eine mittels TLS/SSL geschützte Verbindung aufgebaut werden soll, wird diese Verbindung zwischen Proxy und dem entsprechenden Server des Service Providers im Internet aufgebaut. Die Ende-zu-Ende-Verschlüsselung besteht also zwischen Proxy und Server, weshalb der Proxy die Daten lesen und auch cachen kann. Die Verbindung zwischen Client und Proxy kann – unabhängig von der dahinter liegenden Verbindung – verschlüsselt oder unverschlüsselt sein.

Sollen verschlüsselte Daten auf diese Weise zwischengespeichert werden, ist die Privatsphäre des Clients und damit sein Recht auf informationelle Selbstbestimmung gefährdet,

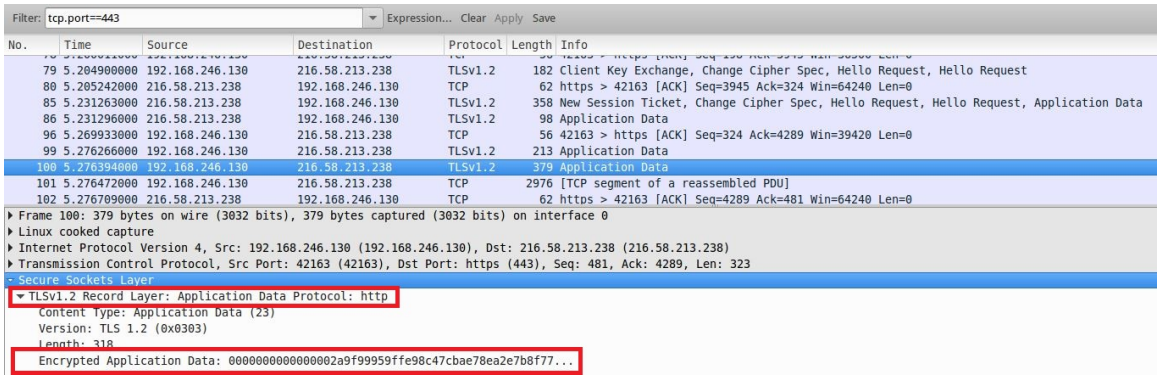

Abbildung 4: Mittels SSL/TLS übertragene Pakete

weshalb ein solcher Proxy nicht ohne weiteres für einen Client verpflichtend in seine Internetverbindung eingebaut werden kann.

Es wäre ggf. möglich, einen Nutzer über ein Opt-In Verfahren an einen solchen Proxy anzubinden. Ein für die Privatsphäre unkritischer Bereich wäre ein Proxy-Server im lokalen Netzwerk des Nutzers, d.h. auf Ebene der End Entities oder Local Gateways. Hierdurch werden die Daten auf vom Nutzer physisch kontrollierten Geräten vorgehalten; ein Zugriff von außerhalb durch Dritte ist hierdurch prinzipiell erschwert. Hieraus ergeben sich auch die in Kapitel Caching im Allgemeinen diskutierten Vor- und Nachteile bezüglich der Entlastung des Netzwerks und Performancesteigerung.

Eine weitere Möglichkeit, Bandbreitennutzung bei TLS-gesicherten Verbindungen einzusparen, ist das in [20] beschriebene Verfahren zur Speicherung von Session-Informationen. Es wird eine Einsparung der Bandbreitennutzung während des TLS-Handshakes von bis zu 72% angegeben. An dieser Stelle sei jedoch angemerkt, dass der TLS-Handshake lediglich einen marginalen Anteil der übertragenen Daten darstellt und die verschlüsselten Daten weiterhin nicht gecacht werden können.

V. Methodologie

In diesem Kapitel wird ein Überblick der Rahmenbedingungen des Experiments gegeben. Hierzu werden zunächst die inhaltlichen Ziele des Experiments definiert und in Zusammenhang mit der zuvor erläuterten Theorie gebracht. Anschließend wird der Versuchsaufbau sowie die geplante Auswertung der durch das Experiment gewonnenen Daten beschrieben.

A. Ziele des Experiments

Untersucht wird ein Cache auf Ebene E.2 (Local Gateway), der im Netzlabor der Hochschule Bonn-Rhein-Sieg aufgestellt wird. Zur Beurteilung der Leistungsfähigkeit eines Caches dieser Ebene werden als wesentliche Merkmale die Hit Ratios bezogen auf die erhaltenen Requests sowie den verursachten Datenverkehr herangezogen. Diese Größen werden im Kontext der Gesamtmenge aller gecachten Daten betrachtet (Cachgröße), da diese den zu leistenden Aufwand zur Bandbreiteneinsparung darstellen. Des Weiteren sollen die Hit Ratios nicht nur über alle Requests bzw. den kompletten Datenverkehr, sondern ebenfalls einzeln aufgeschlüsselt in mehreren Kategorien betrachtet werden. Diese sind Inhalts-Typen, wie sie in Kapitel Inhalts-Typen betrachtet wurden. Ein besonderes Augenmerk soll zudem auf verschlüsselte Inhalte gelegt werden, um nicht nur Aussagen zur Cachbarkeit der Inhalts-Typen an sich treffen zu können, sondern ebenfalls hinsichtlich der verschlüsselten Übertragungsweise (siehe Kapitel Caching im Kontext der Kryptographie). Diese Vorgehensweise ermöglicht es, die in dem Kapitel Strategische Objekte erarbeiteten Theorien praktisch zu verifizieren.

B. Versuchsaufbau

Der Cache wird realisiert durch einen Squid Server [19], welcher einen Hybrid aus Cache- und Proxy-Server darstellt. Dieser bietet zahlreiche Konfigurationsmöglichkeiten; eine Auswahl relevanter, im Rahmen des Experiments vorgenommener Einstellungen wird im Folgenden näher beschrieben.

Cache Directory Size: Beschreibung der Größe des Cachedirectories. Diese wurde auf 90,000 MiB festgesetzt, um einen Überlauf des Caches zu vermeiden und somit unabhängig von den variierenden Kosten des Cachens in Produktivumgebungen Aussagen treffen zu können.

Cache Memory Size: Maximale kumulierte Größe aller im Arbeitsspeicher vorgehaltenen Daten. Zur Durchführung des Experiments wurde diese Einstellung auf 2048 MiB festgelegt.

Cache Replacement Policy / Memory Replacement Policy: Cache-Ersetzungs-Strategie für die auf dem lokalen Massenspeicher bzw. im Arbeitsspeicher vorgehalten Daten. Beide Einstellungen wurden auf Least-Recently-Used (LRU) eingestellt. Durch die eingestellte Cachegröße verliert diese Option jedoch an Bedeutung für das Experiment.

Minimum / Maximum Object Size: Angabe des Rahmens, in welchem Objekte liegen müssen, um gecacht zu werden. Objekte, deren Größe Minimum Object Size unterschreitet bzw. Maximum Object Size überschreitet, werden weder im Arbeits- noch Massenspeicher vorgehalten. Es sind für dieses Experiment diesbezüglich

jedoch keine Einschränkungen festgelegt worden, um ein authentisches Ergebnis bei der Betrachtung der Hit Ratios in Bezug auf die Objektgröße zu erhalten und festellen zu können, ob und wie sich die Objektgröße auf die Bandbreitenentlastung auswirkt.

Offline Mode: Diese Option kann entweder aktiviert oder deaktiviert werden. Im aktivierten Zustand sorgt sie dafür, dass Squid niemals gecachte Objekte validieren wird, d.h. die Aktualität verifiziert. Diese Option ist Rahmes des Versuchsaufbaus deaktiviert.

C. Auswertung der Daten

Sämtliche eingehende Requests sowie mit ihnen verknüpfte Informationen werden durch Squid in einer Log-Datei festgehalten. Aus dieser Datenbasis werden durch den Loganalyzer Calamaris [21] Reports generiert, welche die im Abschnitt Ziele des Experiments beschriebenen statistischen Daten erfassen. Eine Auflistung weiterer alternativer Analysetools ist unter [22] zu finden. Ein solcher Report wird einmal täglich automatisiert aus der *gesamten* Logdatei erstellt sowie zusätzlich die Größe des Cache Directories gemessen. Die Logdatei wird anschließend nicht geleert, so dass bereits beantwortete Requests weiterhin in die gemessenen Statistiken einfließen.

Als weiteres Werkzeug zur Beurteilung des Caches wird eine Reihe von Kennzahlen aus den Daten des Reports generiert und deren Entwicklung über den gesamten Versuchszeitraum hinweg betrachtet. Diese Kennzahlen sind im Folgenden beschrieben:

1) Haupt-Kenzahlen: Die Kennzahlen in diesem Bereich geben einen kurzen Gesamtüberblick über die Leistung des Caches.

Hit Ratio: Gibt in Prozent an, wie viele der eingegangenen Requests durch im Cache vorgehaltene Objekte bedient werden konnten.

$$\frac{Aus\ Cache\ beantwortete\ Requests}{Anzahl\ aller\ Requests} \cdot 100\% \quad (1)$$

Effectivity: Das Verhältnis zwischen aus dem Cache beantwortetem Datenvolumen und dem gesamten, an die Nutzer gesendeten Datenvolumen gibt prozentual die Bandbreitenersparnis an.

$$\frac{Aus\ Cache\ beantwortetes\ Datenvolumen}{Insgesagmt\ angefragtes\ Datenvolumen} \cdot 100\% \quad (2)$$

Efficiency: Vergleicht den eingesparten Datenverkehr (Nutzen des Caches) mit der Menge der gespeicherten Objekte (Kosten des Caches).

$$\frac{Aus\ Cache\ beantwortetes\ Datenvolumen}{Groesse\ aller\ im\ Cache\ vorhandenen\ Daten} \cdot 100\% \quad (3)$$

Cacheability: Anteil der cachebaren Daten am gesamten eingehenden Datenverkehr. Aus dem Cache beantwortete Anfragen werden hierbei nicht berücksichtigt, da andernfalls das dahinterstehende Datenvolumen mehrfach gezählt würde.

$$\frac{Groesse\ aller\ gecachten\ Daten}{Eingehendes\ Datenvolumen} \cdot 100\% \quad (4)$$

mit

$$\begin{aligned} &Eingehendes\ Datenvolumen \\ &= Insgesagmt\ angefragtes\ Datenvolumen \\ &- Aus\ Cache\ beantwortetes\ Datenvolumen \end{aligned} \quad (5)$$

2) Kennzahlen für Inhalts-Typen: Kennzahlen in diesem Bereich schlüsseln die generelle Hit Ratio auf, um herauszustellen, welche Inhalts-Typen in besonderem Maße cachbar bzw. nicht cachbar sind und welche Ergebnisse die verschlüsselten Inhalte liefern konnten. Hierbei werden die Folgenden betrachtet:

Image Hit Ratio: Die Hit Ratio bezogen auf alle Bildformate wie JPG, GIF, usw.

Text Hit Ratio: Die Hit Ratio bezogen auf alle Textformate, z.B. Plain, XML, HTML.

Video Hit Ratio: Die Hit Ratio bezogen auf Videoformate.

Application Hit Ratio: Die Hit Ratio aller Applikations-Daten. In diesen Bereich fallen beispielsweise im JSON-Format übertragene Daten.

Secure Hit Ratio: Die Hit Ratio bezogen auf jeglichen verschlüsselten Datenverkehr.

Die Berechnung dieser Kennzahlen erfolgt analog zur Berechnung der Hit Ratio im vorherigen Abschnitt Haupt-Kenzahlen. Da kein Anteil von Audio-Inhalten am übertragenen Datenvolumen bzw. den gestellten Requests gemessen werden konnte (ähnlich wie bei Ramanan et al. [8]), werden diese von der Betrachtung ausgeschlossen.

VI. Ergebnisse & Interpretation

An dem zehn Tage andauernden Experiment konnten die Daten von 14 Teilnehmern erhoben werden. Über den Versuchszeitraum hinweg wurden insgesamt 214.390 Requests, welche in einem Gesamtdatenvolumen von 30.738 MiB resultieren, über den Cache gesendet. Von der Gesamtzahl der Versuchsteilnehmer hatten neun einen Anteil von über einem Prozent (307,38 MiB) am Gesamtdatenvolumen; die Gesamtgröße der im Cache vorgehaltenen Objekte betrug am letzten Tag des Experiments 5.537 MiB.

Die in diesem Kapitel präsentierten Liniendiagramme geben eine kumulierte Übersicht über den Verlauf der erfassten Daten. D.h. sie geben keine Auskunft über die Ergebnisse des jeweiligen Tages, sondern beinhalten ebenfalls die Ergebnisse der vorigen Tage.

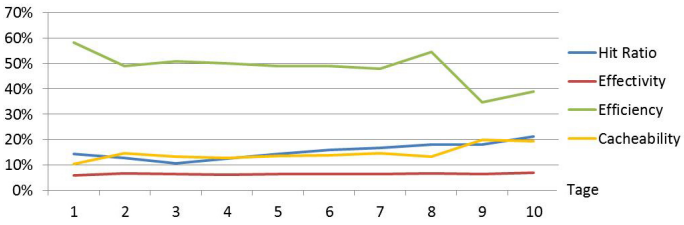

Abbildung 5: Kumulierter Verlauf der Hauptkennzahlen (Eigene Abb.)

Wie aus Abbildung 5 ersichtlich wird, schwankten die aus dem Cache beantworteten Requests (Hit Ratio) zwischen zehn Prozent und leicht über 20%, wohingegen die ersparte Bandbreite (Effectivity) konstant bei ca. sechs bis sieben Prozent lag.

Eine Auffälligkeit ist der Rückgang der ansonsten konstant bei 50% liegenden Efficiency um ca. 20 Prozentpunkte von Tag acht auf Tag neun. Dieser lässt sich durch eine einzelne Anfrage auf ein zwei GiB großes Objekt erklären, aus welcher sich folglich keine Bandbreitenersparnis ergeben kann.

Aus demselben Grund steigt der Anteil des cachebaren Datenvolumens (*Cacheability*) zum gleichen Zeitpunkt von ca. 13% auf knapp 20%.

Die am Ende des Versuchs durchschnittlich ermittelten sieben Prozent Bandbreitenersparnis lassen sich wie in Abbildung 6 dargestellt nach Objektgrößen aufschlüsseln:

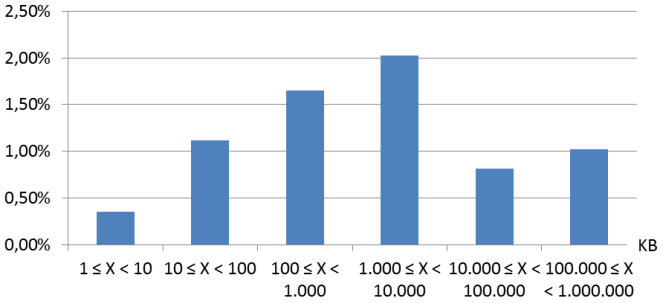

Abbildung 6: Eingesparte Bandbreite nach Objektgröße (Eigene Abb.)

Den höchsten Anteil an der Bandbreitenersparnis haben mit zwei Prozent die Objekte der Größe von ein bis zehn MB, gefolgt von den Objektgrößen 100 KB bis ein MB mit 1,64%. Eine ähnliche Effektivität von jeweils ungefähr einem Prozent weisen die Objekte mit den Größen zehn KB bis 100 KB und 100 MB bis 1.000 MB auf. Ebenso kann es lohnenswert sein, Objekte von einem bis zehn KB und zehn bis 100 MB im Cache zwischenzuspeichern.

Diese Ergebnisse zeigen deutlich, dass es nicht effektiv ist, kleinere Objekte als ein KB und größere Objekte als ein GB in den Cache aufzunehmen. Letzteres ist insbesondere durch den zuvor beschriebenen Rückgang in der Effizienz belegt.

In der zum Thema Caching veröffentlichten Literatur werden die beantworteten Requests, häufig aber nicht die dadurch eingesparte Bandbreite betrachtet. Zur besseren Vergleichbarkeit werden bei der Betrachtung der Inhalts-Typen deshalb die Hit Ratios in den Vordergrund gerückt.

In Abbildung 7 werden die prozentual vom Cache beantworteten Requests aufgeschlüsselt nach Inhalts-Typen aufgezeigt. Mit Ausnahme der Hit Ratio des angefragten Video-Inhaltes schwankten diese über den Versuchszeitraum hinweg recht stark.

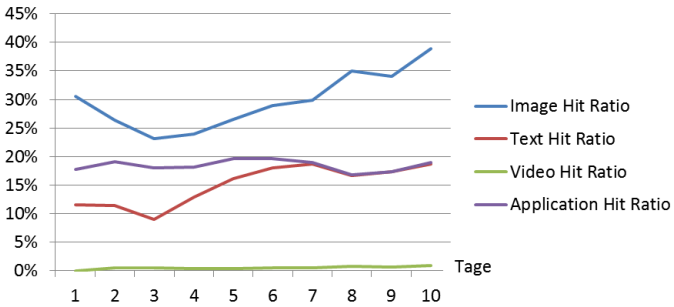

Abbildung 7: Kumulierter Verlauf der Inhalts-Typen (Eigene Abb.)

Aus den Ergebnissen des Experimentes geht hierbei aufgrund der geringen Hit Ratio eine schlechte Cachbarkeit von Video-Inhalten hervor. Zudem lassen sich lediglich 1,2% aller Requests auf Video-Inhalte zurückführen. Diese 1,2% der Requests machen jedoch knapp 17% des Datenvolumens aus. Hieraus folgt, dass sich ein beträchtlicher Teil des heute aufkommenden Datenaufkommen nur schwer reduzieren lässt.

Einen ähnlich hohen Anteil am Datenvolumen haben mit 16,19% die übertragenen Bilddateien. Von diesen ließen sich jedoch insgesamt knapp 40% der Requests aus dem Cache bedienen, allerdings erst nach einer „Lern-Zeit“, während der die später erneut angefragten Bilder im Cache abgespeichert wurden.

Im Schnitt konnte zwar jede fünfte Anfrage auf Applikations- und Textdaten aus dem Cache beantwortet werden. Durch den verschwindend geringen Anteil solcher Daten an der Bandbreitennutzung (jeweils 1,2%) ergibt sich hier allerdings kein nennenswerter Vorteil.

Verschlüsselte Inhalte ließen sich – wie in Kapitel Caching im Kontext der Kryptographie bereits herausgearbeitet – zu keinem Zeitpunkt im Cache wiederverwerten. Hier gehen aufgrund der erhobenen Daten 45% des entstandenen Datenvolumens in die nicht cachebare und damit nicht vermeidbare Bandbreitennutzung ein. Wie bereits beschrieben ließen sich zudem lediglich 20% des eingegangenen Datenvolumens cachen. Somit ist leicht ersichtlich, dass zusätzlich zu den verschlüsselt übertragenen Daten ebenfalls 35% der gesamten Daten von Squid nicht gecacht werden konnten.

Dies wird durch nachfolgendes Diagramm nochmals zusammengefasst:

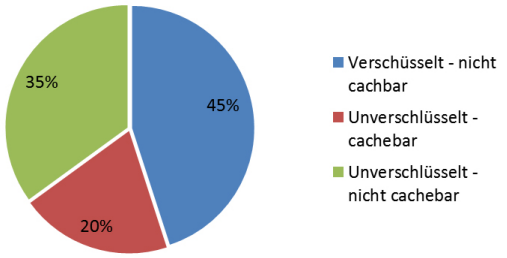

Abbildung 8: Zusammensetzung des Datenverkehrs (Eigene Abb.)

VII. Zusammenfassung und Fazit

Das Ziel dieser Seminararbeit bestand darin, den Stand der Technik von Caching herauszuarbeiten sowie die Reduktion der Bandbreitennutzung durch den Einsatz einer solchen Technik zu quantifizieren. Hierbei sollte ebenfalls die Problematik mit verschlüsselten Daten adressiert und angegangen werden.

Um diese Problemstellung angemessen zu bearbeiten, wurden zu Beginn die theoretische Grundlagen und Ideen behandelt, welche im Abschluss durch ein praktisches Experiment bestätigt werden konnten.

Im Zuge des theoretischen Teils wurde zu Beginn eine einheitliche Terminologie entwickelt, die dem Leser ungeachtet der eingesetzten Technologie eine Vergleichbarkeit der Publikationen auf diesem Bereich ermöglicht. Aufbauend auf dieser wurde die generische Arbeitsweise eines Caches beschrieben und etwaige allgemeine Vor- und Nachteile diskutiert. Es zeigte sich, dass durch Caching eine höhere Geschwindigkeit bei der Diensterbringung erzielt werden kann, die zugleich geringere Pfadkosten nach sich zieht, wobei jedoch Probleme mit der Datenkonsistenz entstehen können. Zudem wurde ersichtlich, dass die genannten Vor- und Nachteile in unmittelbarer Abhängigkeit einer strategisch guten Platzierung des Caches stehen: Ein Cache entlastet in besonderer Weise den direkt hinter ihm liegenden Link (Pfadkostenersparnis); wird der Cache hingegen in der Nähe der End-Entity bzw. dem Nutzer platziert, so können sich erhebliche Performance-Vorteile für diesen ergeben.

Im nächsten Schritt wurde betrachtet, welche Datenobjekte lohnenswert sind gecacht zu werden. Bild-Inhalte von prominenten Inhalts-Anbietern wurden hierbei als besonders sinnvoll erachtet, während die Hypothese aufgestellt werden konnte, dass verschlüsselte Inhalte aufgrund der Kapselung bei SSL/TLS (ohne weiteres) nicht gecacht werden können. Zudem wurden Caching-Architekturen vorgestellt, die eine Zusammenarbeit verschiedener Caches erlauben sowie drei grundlegende Strategien vorgestellt, wie Caching von der technischen Seite her umgesetzt werden kann: Die Erfassungs-Strategie, die Daten-Organisations-Strategie und zuletzt die Cache-Ersetzungs-Strategie. Bei den Cache Erfassungs-Strategien zeigte sich, dass ganz besonders das *proaktive Caching* in der aktuellen Literatur zunehmend an Bedeutung gewinnt und zukünftig auch im praktischen Einsatz mehr Relevanz erfahren könnte.

Aufbauend auf der eingangs erläuterten Problemstellung sowie den herausgearbeiteten Grundlagen und Ideen wurden abschließend Kennzahlen abgeleitet, die diese aus praktischer Sicht anhand eines Experimentes bestätigen sollten. Hierbei zeigte sich, dass die von Ramnan et al. in [8] präsentierten Ergebnisse zur Cachebarkeit unterschiedlicher Inhalts-Typen durch das durchgeführte Experiment größtenteils nachvollzogen und bestätigt werden konnten: Der Anteil der Requests für Bild-Inhalte war der höchste und konnte mit Abstand am besten vom Cache wiederverwendet werden. Text-Inhalte hatten zwar eine geringere, aber dennoch gute Hit Ratio, allerdings ist die mögliche Ersparnis bei der Bandbreitennutzung durch den geringen Anteil am Datenvolumen vernachlässigbar. Auch der Video-Inhalt hatte einen geringen Anteil an den verarbeiteten Requests, verursachte jedoch den höchsten Datenverkehr aller Inhalts-Typen. Auswirkungen auf die genutzte Bandbreite hatte das Cachen ebendieser aufgrund der schlechten Hit Ratio allerdings kaum.

Anfragen im Rahmen verschlüsselter Verbindungen (SSL/TLS) konnten zu keinem Zeitpunkt aus dem Cache beantwortet werden und bestätigten somit die aufgestellte Hypothese. Während des Experiments konnte ein Anteil von 45% verschlüsselt übertragener Objekte am Gesamtaufkommen des Datenverkehrs gemessen werden. Hierdurch wird die Effektivität eines von mehreren Usern gemeinsam genutzten Caches beeinträchtigt. Zur Lösung dieser Problematik sollte dem Cache die Möglichkeit gegeben werden, als Man-in-the-Middle verschlüsselte Objekte einsehen und zwischenspeichern zu können sowie zukünftige Anfragen auf dieselben Objekte als solche erkennen zu können. Dieser Lösungsansatz geht jedoch mit weitreichenden Konsequenzen für Datensicherheit, Privatsphäre und das Recht auf informationelle Selbstbestimmung der Anwender einher.
Die von Ramanan et al. in [8] lediglich gemessenen 26,1% an nicht-cachbarem Datenaufkommen konnten in diesem Experiment allerdings nicht nachvollzogen werden; hier fielen 80% in die Kategorie nicht-cachbares Datenvolumen, wobei mit bereits 45% die verschlüsselten Daten einen enormen Beitrag leisteten.

Insgesamt konnte über den gesamten Zeitverlauf eine Bandbreitenreduktion von knapp sieben Prozent erzielt werden; dieser Wert deckt sich mit den von Ramanan et al. gemessenen 6,63% [8].

Eine allgemeine Möglichkeit die Effektivität eines Caches weiter zu steigern und Lastspitzen auszugleichen besteht in der Anwendung von *proaktivem Caching*. Durch die Erstellung von Nutzerprofilen kann der Cache Daten laden, bevor diese tatsächlich vom Anwender angefragt werden.

Literatur

[1] Cisco Systems (Hrsg.). Cisco-studie: Mobiler datenverkehr wächst bis 2018 weltweit um das 11-fache. http://globalnewsroom.cisco.com/de/de/press-releases/cisco-studie-mobiler-datenverkehr-wachst-bis-2018-nasdaq-csco-1087573. [Online; Stand 01.12.2015].

[2] McKinsey (Hrsg.). The internet of things: Mapping the value beyond the hype. http://www.mckinsey.de/sites/mck_files/files/unlocking_the_potential_of_the_internet_of_things_full_report.pdf. [Online; Stand 26.11.2015].

[3] X. Cai; S. Zhang; Y. Zhang. Economic analysis of cache location in mobile network. In *Wireless Communications and Networking Conference (WCNC), 2013 IEEE*, pages 1243 – 1248, 2013.

[4] X. Wang; M. Chen; T. Taleb; A. Ksentini; V. Leung. Cache in the air: exploiting content caching and delivery techniques for 5g systems. In *Communications Magazine, IEEE*, volume 52, pages 131 – 139, 2014.

[5] E. Bastug; M. Bennis; M. Debbah. Social and spatial proactive caching for mobile data offloading. In *Communications Workshops (ICC), 2014 IEEE International Conference*, 2014.

[6] J. Costa-Requena; R. Kantola; J. Llorente; V. Ferrer; J. Manner; A.Y. Ding; L. Yanhe; S. Tarkoma. Software defined 5g mobile backhaul. In *5G for Ubiquitous Connectivity (5GU), 2014 1st International Conference*, pages 258 – 263, 2014.

[7] E. Bastug; M. Bennis; M. Debbah. Living on the edge: The role of proactive caching in 5g wireless networks. *Communications Magazine, IEEE*, 52(8):82–89, Aug 2014.

[8] B. A. Ramanan; L.M. Drabeck; M. Haner; N. Nithi; T.E. Klein; C. Sawkar. Cacheability analysis of http traffic in an operational lte network. In *Wireless Telecommunications Symposium (WTS), 2013*, pages 1 – 8, 2013.

[9] K. Doppler; M. Rinne; C. Wijting; C.B. Ribeiro; K. Hugl. Device-to-device communication as an underlay to lte-advanced networks. In *Communications Magazine, IEEE*, volume 47, pages 42 – 49, 2009.

[10] Cisco Systems (Hrsg.). Network caching technologies. http://docwiki.cisco.com/wiki/Network_Caching_Technologies. [Online; Stand 26.11.2015].

[11] Heriot-Watt University (Hrsg.). Pros and cons of web caching. http://www.macs.hw.ac.uk/%7ehamish/3NI/topic172.html. [Online; Stand 25.11.2015].

[12] R. Fielding; UC Irvine; J. Gettys; J. Mogul; H. Frystyk; L. Masinter; P. Leach; T. Berners-Lee. RFC 2616. https://tools.ietf.org/html/rfc2616. [Online; Stand 01.12.2015].

[13] M. Nottingham; J. Mogul. RFC 4229. https://tools.ietf.org/html/rfc4229. [Online; Stand 01.12.2015].

[14] I. Grigorik. Http caching. https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/http-caching. [Online; Stand 05.12.2015].

[15] P. Rodriguez; C. Spanner; E.W. Biersack. Analysis of web caching architectures: hierarchical and distributed caching. *Networking, IEEE/ACM Transactions on*, 9(4):404–418, Aug 2001.

[16] G. Barish; K. Obraczke. World wide web caching: trends and techniques. *Communications Magazine, IEEE*, 38(5):178–184, May 2000.

[17] M. Liu; F. Wang; D. Zeng; L. Yang. An overview of world wide web caching. In *Systems, Man, and Cybernetics, 2001 IEEE International Conference on*, volume 5, pages 3045–3050 vol.5, 2001.

[18] S. Woo; E. Jeong; S. Park; J. Lee; S. Ihm K. Park. Comparison of caching strategies in modern cellular backhaul networks. In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '13, pages 319–332, New York, NY, USA, 2013. ACM.

[19] squid-cache.org (Hrsg.). Squid: Optimising web delivery. http://www.squid-cache.org/. [Online; Stand 07.12.2015].

[20] H. Shacham; D. Boneh; E. Rescorla. Client-side caching for tls. *ACM Trans. Inf. Syst. Secur.*, 7(4):553–575, November 2004.

[21] Cord Beermann. Calamaris (english). http://cord.de/calamaris-english. [Online; Stand 07.12.2015].

[22] squid-cache.org (Hrsg.). Squid related log analysis software. http://www.squid-cache.org/Misc/log-analysis.html. [Online; Stand 10.12.2015].