Network Architectures
and Services
NET 2010-08-2

FI & IITM SS 2010

Proceedings of the Seminars Future Internet (FI) and Innovative Internet Technologies and Mobile Communications (IITM)

Summer Semester 2010

Munich, Germany, 15.04.-23.07.2010

Editors

Georg Carle, Corinna Schmitt

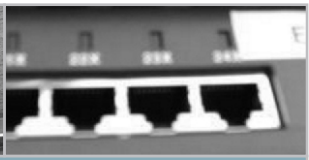
Organisation

Chair for Network Architectures and Services
Department of Computer Science, Technische Universität München

Technische Universität München







Network Architectures
and Services
NET 2010-08-2

FI & IITM

SS 2010

Proceedings zu den Seminaren
Future Internet (FI) und
Innovative Internet-Technologien und Mobilkommunikation (IITM)

Sommersemester 2010

München, 15.04.-23.07.2010

Editoren: Georg Carle, Corinna Schmitt

Organisiert durch den Lehrstuhl Netzarchitekturen und Netzdienste (I8),
Fakultät für Informatik, Technische Universität München

Technische Universität München

Seminar FI & IITM SS 2010
Seminar: Future Internet
Seminar: Innovative Internet-Technologien und Mobilkommunikation,
Sommersemester 2010

Editors:

Georg Carle
Lehrstuhl Netzarchitekturen und Netzdienste (I8)
Technische Universität München
D-85748 Garching b. München, Germany
E-mail: carle@net.in.tum.de
Internet: http://www.net.in.tum.de/~carle/

Corinna Schmitt
Lehrstuhl Netzarchitekturen und Netzdienste (I8)
Technische Universität München
D-85748 Garching b. München, Germany
E-mail: schmitt@net.in.tum.de
Internet: http://www.net.in.tum.de/~schmitt/

Cataloging-in-Publication Data

Seminar FI & IITM SS2010
Proceedings zum Seminar Future Internet
München, Germany, 15.04.-23.07.2010
Georg Carle, Corinna Schmitt
ISBN: 3-937201-11-4

ISSN: 1868-2634 (print)
ISSN: 1868-2642 (electronic)
Lehrstuhl Netzarchitekturen und Netzdienste (I8) NET 2010-08-2
Series Editor: Georg Carle, Technische Universität München, Germany
© 2010, Technische Universität München, Germany

Vorwort

Wir präsentieren Ihnen hiermit die Proceedings zu den Seminaren „Future Internet“ (FI) und “Innovative Internettechnologien und Mobilkommunikation” (IITM), die im Sommersemester 2010 an der Fakultät Informatik der Technischen Universität München stattfanden.
Im Seminar FI wurden Beiträge zu unterschiedlichen Fragestellungen aus den Gebieten Internettechnologien und Mobilkommunikation vorgestellt. Die folgenden Themenbereiche wurden abgedeckt:

- NFC
- WiMAX
- Network Resilience
- Transition to IPv6
- High Speed Network Monitoring
- Purpose and security analysis of RASTER
- Google Big Table
- Stream Control Transmission Protocol (SCTP)
- Next Generation Networks

Im Seminar IITM wurden Vorträge zu verschiedenen Themen im Forschungsbereich Internet-Technologien und Mobilkommunikation vorgestellt. Die folgenden Themenbereiche wurden abgedeckt:

- DDoS-Angriffe. Und Verteidigungsstrategien
- Stärken und Schwächen von PKI
- Verteilte PKI in P2P Netzwerken
- Schwächen der aktuellen Internetarchitektur
- Traceroute Zoo
- Mercator
- MapReduce
- Van Jacobsons CCN
- OpenFlow

Wir hoffen, dass Sie den Beiträgen dieser Seminare wertvolle Anregungen entnehmen können. Falls Sie weiteres Interesse an unseren Arbeiten habe, so finden Sie weitere Informationen auf unserer Homepage http://www.net.in.tum.de.

München, August 2010

Georg Carle

Corinna Schmitt

Preface

We are very pleased to present you the interesting program of our main seminars on “Future Internet” (FI) and “Innovative Internet Technologies and Mobil Communication” (IITM) which took place in the summer semester 2010.

In the seminar FI we deal with issues of Future Internet. The seminar language was German, and the majority of the seminar papers are also in German. The following topics are covered by this seminar:

- NFC
- WiMAX
- Network Resilience
- Transition to IPv6
- High Speed Network Monitoring
- Purpose and security analysis of RASTER
- Google Big Table
- Stream Control Transmission Protocol (SCTP)
- Next Generation Networks

In the seminar IITM talks to different topics in innovate Internet technologies and mobile communications were presented. The seminar language was German, and also the seminar papers. The following topics are covered by this seminar:

- DDoS-Attacks and Defence Possibilities
- Strengths and Weaknesses of PKI
- Distributed PKI in P2P Networks
- Weaknesses of today’s Internet achitecture
- Traceroute Zoo
- Mercator
- MapReduce
- Van Jacobsons CCN
- OpenFlow

We hope that you appreciate the contributions of these seminars. If you are interested in further information about our work, please visit our homepage http://www.net.in.tum.de.

Munich, August 2010

Seminarveranstalter

Lehrstuhlinhaber

Georg Carle, Technische Universität München, Germany

Seminarleitung

Corinna Schmitt, Technische Universität München, Germany

Betreuer

Tobias Bandh, *Technische Universität München, Wiss. Mitarbeiter I8*
Nathan Evans, *Technische Universität München, DFG Emmy Noether Research Group Member*
Marc Fouquet, *Technische Universität München, Wiss. Mitarbeiter I8*
Christian Grothoff, *Technische Universität München, DFG Emmy Noether Research Group Leader*
Dirk Haage, *Technische Universität München, Wiss. Mitarbeiter I8*
Ralph Holz, *Technische Universität München, Wiss. Mitarbeiter I8*
Nils Kammenhuber, *Technische Universität München, Wiss. Mitarbeiter I8*
Holger Kinkelin, *Technische Universität München, Wiss. Mitarbeiter I8*
Andreas Müller, *Technische Universität München, Wiss. Mitarbeiter I8*
Heiko Niedermayer, Technische Universität München, Wiss. Mitarbeiter I8
Marc-Oliver Pahl, *Technische Universität München, Wiss. Mitarbeiter I8*
Johann Schlamp, *Technische Universität München, Wiss. Mitarbeiter I8*
Matthias Wachs, *Technische Universität München, DFG Emmy Noether Research Group Member*

Kontakt:

{carle,schmitt,bandh,hirvi,pahl,niedermayer,evans,grothoff,mueller,fouquet, schlamp, haage, kinkelin, wachs, holz}@net.in.tum.de

Seminarhomepage

http://www.net.in.tum.de/de/lehre/ss10/seminare/

Inhaltsverzeichnis

Seminar Future Internet

Session 1: Mobilkommunikation

Session 2: Internet

Session 3: Monitoring und Routing

Seminar Innovative Internettechnologien und Mobilkommunikation

Session 1: Sicherheit

Session 2: Strukturanalyse und Datenverarbeitung

Session 3: Konzepte für „Future Internet“

Near Field Communication

Andrea Cuno
Betreuer: Tobias Bandh
Seminar Future Internet SS2010
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: cunoa@in.tum.de

KURZFASSUNG

Near Field Communication (NFC) ist die Bezeichnung für eine neue drahtlose Technologie, die die Kommunikation zwischen zwei Geräten ermöglicht, wobei die Distanz zwischen diesen wenige Zentimeter beträgt. Die Funktionsweise von NFC baut auf anderen, bereits genutzten Technologien auf. Dabei unterstützt der NFC-Standard verschiedene Kommunikationsformen. Dies trägt dazu bei, dass die Technologie in vielen unterschiedlichen Bereichen zum Einsatz kommen kann. Das Bezahlen mit NFC-fähigen Handys, Ticketing und schnelle, einfache Informationsübertragung sind nur einige Anwendungsfelder.

Schlüsselworte

NFC, RFID, Smart Card, Tags, Peer-to-Peer-Kommunikation

1. EINLEITUNG

Die 2002 gemeinsam von Sony und NXP Semiconductors entwickelte Near Field Communication ermöglicht eine drahtlose Verbindung zwischen zwei Geräten oder Gegenständen, indem sie nahe aneinander gehalten werden. Dabei ermöglicht Near Field Communication sowohl lesenden, als auch schreibenden Zugriff. Die Datenübertragung bei NFC erfolgt auf einer festgelegten Frequenz und die Datenübertragungsrate beträgt derzeit bis zu 424 kbit/s. Zwischen den kommunizierenden Geräten besteht eine Entfernung von circa 0 bis 10 cm. Sobald zwei NFC-fähige Geräte sich innerhalb dieses Abstands befinden, wird die Verbindung hergestellt. Die geringe Distanz erschwert ungewolltes Abhören, was zur Sicherheit für die Datenübertragung beiträgt.
Die neue Technologie basiert auf bereits großflächig genutzten Techniken wie RFID und Smart Cards. Ein Teil der heutigen Mobiltelefone ist schon mit NFC-Support ausgestattet. Zudem werden verschiedene Anwendungen in Pilotprojekten verwirklicht und getestet.
Im Folgenden werden zunächst die grundlegenden Technologien, auf denen NFC aufbaut, vorgestellt, sowie die technische Funktionsweise der Near Field Communication erläutert. Anschließend werden die wichtigsten Anwendungsgebiete aufgezeigt, sowie einige konkrete Anwendungsbeispiele ausführlicher behandelt. Im vierten Abschnitt folgt eine kurze Zusammenfassung und ein Ausblick.

2. WIE IST NFC TECHNISCH REALISIERT?

Die Near Field Communication baut auf anderen Technologien auf, hat aber eigene Schlüsselmerkmale, die sie von diesen Technologien abgrenzen.

2.1 Eingesetzte Technologien

Die technische Realisierung der Datenübertragung bei NFC basiert auf der so genannten RFID-Technologie (Radio Frequency Identification). Um einen sicheren Datenaustausch zu ermöglichen, sind in der NFC-Technologie zudem Sicherheitsmaßnahmen integriert, die bereits in Smart Cards eingesetzt werden.

2.1.1 Radio Frequency Identification - RFID

RFID ist eine Technik für die kontaktlose Identifikation beziehungsweise für die Datenerfassung und -speicherung.
Die Datenübertragung bei RFID erfolgt, indem ein elektromagnetisches Feld erzeugt wird.
In einem RFID-System existieren so genannte Tags und Lesegeräte. Ein Lesegerät dient dazu, Daten, die auf einem Tag gespeichert sind, zu lesen und gegebenenfalls auch auf den Tag zu schreiben. Für eine weitere Auswertung können die Daten, die das Lesegerät erfasst hat, weitergeleitet werden.
Tags sind RFID-Transponder, die zum Beispiel an einem Produkt befestigt sind. Sie bestehen aus einem Mikrochip und einer Spule als Antenne (siehe Abb. 1). Es existieren

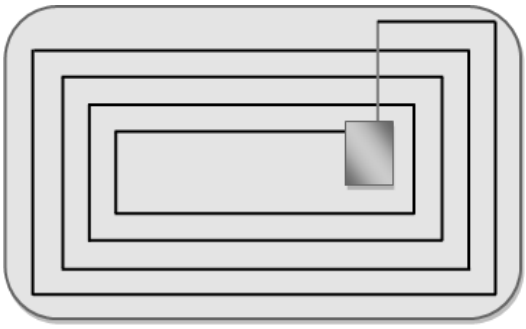


Abbildung 1: RFID-Tag mit Spule und Chip

zwei verschiedene Arten von Tags, wobei der wesentliche Unterschied in der Energieversorgung der Transponder besteht. Bei aktiven Tags stellt diese eine kleine eingebaute Batterie

sicher. Passive Tags hingegen besitzen keine eigene Energieversorgung. Stattdessen beziehen sie mithilfe der Spule ihre Energie aus dem elektromagnetischen Feld, das das Lesegerät erstellt.
Bei passiven Tags ist eine maximale Entfernung von circa 120 cm zwischen Lesegerät und Tag gegeben. Mit aktiven RFID-Tags hingegen ist eine Reichweite von 100 m möglich. Zudem ist die mögliche Datenübertragungsrate bei passiven Tags geringer als bei aktiven Tags.
Für die RFID-Technik existieren verschiedene mögliche Frequenzbereiche. Nachfolgend sind einige häufig eingesetzte Frequenzen aufgelistet.

Tabelle 1: RFID-Frequencen

Frequenzbereich	max.Reichweite	Übertragungsrate
125–134 kHz	0,5 m	<1kbit/s
13.56 MHz	1,5 m	25 kbit/s
433 MHz	100 m	30 kbit/s
865 – 956MHz	5 m	30 kbit/s
2.45 GHz	10 m	100 kbit/s

Die Frequenzbereiche unterscheiden sich vor allem in der möglichen Reichweite und der Geschwindigkeit der Datenübertragung [24]. In Abhängigkeit ihrer Eigenschaften werden diese Frequenzen in verschiedenen Anwendungsbereichen von RFID eingesetzt.

2.1.2 Smart Cards

Smart Cards, auch Prozessorchipkarten genannt, sind Chipkarten, die über einen Mikroprozessor, einen Speicher und ein Betriebssystem verfügen.
Üblicherweise gibt es in Smart Cards keine Möglichkeit, direkt auf die gespeicherten Daten auf der Karte zuzugreifen. Stattdessen erfolgt der Zugriff über den Mikroprozessor. Da auf diesem Mikroprozessor verschiedene anwendungsspezifische Programme laufen können, ermöglichen Smart Cards viele unterschiedliche Funktionen, die über einfache Datenspeicherung und Datenzugriff hinausgehen. Zu diesen Funktionen gehören verschiedene Sicherheitsdienste wie Authentifikation, Verschlüsselung, Signatur und Autorisierung. Auch auf der Hardwareebene existieren Vorkehrungen für eine sichere Datenübertragung und Datenspeicherung. Um Manipulationen zu verhindern, besitzen Smart Cards Sensoren, die Änderungen in der Umgebung erkennen. Dazu gehören Temperatur-, Frequenz- und Spannungssensoren. Wenn mittels dieser Sensoren bestimmte Grenzwerte überschritten werden, wird in der Smart Card ein Reset ausgelöst, beziehungsweise die Karte wird unbrauchbar gemacht.
Neben weitergehenden Schutzmaßnahmen gibt es zudem eine Passivierungsschicht auf dem Chip. Sobald diese entfernt wird, führt dies zur Deaktivierung der Karte.

2.2 Funktionsweise von NFC

Die Technik der Datenübertragung baut auf der RFID-Technologie auf. Im Gegensatz zu RFID arbeitet NFC jedoch immer auf einer Frequenz von 13,56 MHz. Diese Frequenz ist unlizenziert und weltweit verfügbar.
Die Reichweite bei der Datenübertragung mittels NFC beträgt circa 0 bis 10 cm. Derzeit liegen die möglichen Übertragungsraten bei 106 kbit/s, 212 kbit/s und 424 kbit/s.

Bei der Beschreibung des Kommunikationsablaufes werden die Begriffe Initiator und Target unterschieden.

- Initiator - das NFC-Gerät, das die Kommunikation startet
- Target - das Gerät, das auf eine Anfrage antwortet

Es existieren zwei verschiedene Verfahren für die Datenübertragung bei NFC, der aktive und der passive Modus.
Im so genannten aktiven Modus erzeugen Initiator und Target jeweils ein eigenes elektromagnetisches Feld, um die Daten zu übertragen.
Bevor der Initiator die Kommunikation startet und sein elektromagnetisches Feld aufbaut, wird geprüft, ob ein anderes Feld in der Umgebung bereits existiert. Falls dies nicht zutrifft, erstellt das Gerät sein eigenes Feld. Dadurch wird das Target, mit dem die Kommunikation stattfinden soll, aktiviert und die Datenübertragung kann stattfinden. Das Senden der Daten erfolgt dabei immer im Wechselbetrieb, das heißt, immer nur jeweils eines der beiden Geräte kann zu einer Zeit senden.
Für die Datenübertragung wird ein Schwingkreis erzeugt. Das bedeutet, zwischen dem im NFC-Gerät eingebauten Kondensator und der Spule wird die Energie abwechselnd umgelagert. Dafür müssen die beiden Komponenten korrekt aufeinander abgestimmt sein. Die Daten werden dann durch Amplitudenmodulation übertragen [5].
Im passiven Modus hingegen erzeugt nur der Initiator ein elektromagnetisches Feld und das Target antwortet, indem es die angeforderten Daten durch Lastmodulation sendet. Als Vorraussetzung für die Lastmodulation muss die Sendefrequenz des Lesegerätes und die Eigenresonanzfrequenz des Tags gleich sein. Lastmodulation bedeutet, dass das Target das elektromagnetische Feld, das der Initiator erzeugt, verändert, indem es dem Feld Energie entzieht [5]. Das kann das Lesegerät registrieren. Durch das Ein- beziehungsweise Ausschalten eines Lastwiderstandes im Tag wird diese Rückwirkung auf das Lesegerät verändert. Wenn nun das Ein- und Ausschalten des Lastwiderstandes durch Daten gesteuert wird, kann das Lesegerät die Daten auf diesem Weg erfassen.

Im Allgemeinen sind Geräte mit NFC-Support in der Lage, zwischen dem aktiven und passiven Modus zu wechseln. Jedes NFC-Gerät ist standardmäßig im passiven Modus. Der passive Modus ist aus Energiespargründen vor allem bei Geräten mit Akku-Betrieb sinnvoll. Zudem kann im passiven Modus selbst dann gesendet werden, wenn das NFC-Gerät ausgeschaltet ist. Das ist zum Beispiel nützlich für Zutrittssysteme, in denen ein NFC-fähiges Mobiltelefon auch im ausgeschalteten Zustand von einem Lesegerät gelesen werden kann und so für den Zugang berechtigt.
Wenn die Anwendung es erfordert, wechselt das Gerät in den aktiven Modus, das heißt, es kann jetzt als Initiator agieren. Neben NFC-fähigen Geräten, die ihren Modus wechseln können, gibt es auch einfache passive NFC-Tags. Diese befinden sich immer im passiven Modus. Sie können gelesen werden, sind jedoch nie Initiator einer Kommunikation mit einem anderen Gerät.

NFC-Geräte können zwischen folgenden standardisierten Kommunikationsarten wechseln (vgl. Abb. 2).

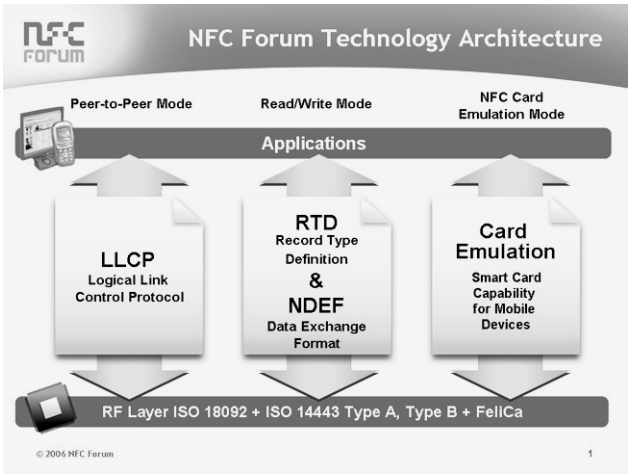


Abbildung 2: NFC-Kommunikationsarten [10]

- Peer-to-Peer Modus - Dieser Modus ermöglicht, dass zwei gleichberechtigte NFC-Geräte Daten miteinander austauschen. Beide Kommunikationspartner befinden sich im aktiven Modus. Dabei wird das Logical Link Control Protocol (LLCP) verwendet. Es unterstützt die bidirektionale Kommunikation zwischen zwei NFC-fähigen Geräten. LLCP spezifiziert zwei Arten der Kommunikation, die verbindungsorientierte und die verbindungslose Kommunikation [18].

- Read/Write-Modus - Im Read/Write-Modus ist das NFC-Gerät im aktiven Modus und kann von allen Tag-Arten lesen, die vom NFC Forum spezifiziert sind. Dabei muss das Datenformat dem NFC Data Exchange Format (NDEF) entsprechen. Das NDEF spezifiziert das Datenformat für die Datenübertragung zwischen NFC-fähigen Geräten und Tags. Die Record Type Definition (RTD) spezifiziert standardisierte Aufzeichnungstypen für die Datenübermittlung.

- NFC Card Emulation Modus - In diesem Modus erscheint das NFC-Gerät für ein externes Lesegerät wie eine kontaktlose Smart Card. Dieser Modus ist wichtig für NFC-Anwendungen, bei denen bereits bestehende Infrastrukturen genutzt werden sollen. Im Gegensatz zum Peer-to-Peer-Modus findet die Kommunikation im Card-Emulation-Modus nicht bidirektional statt. Es befindet sich im passiven Modus und es kann nur von dem Gerät gelesen werden.

Durch diese verschiedenen Modi deckt der NFC-Standard mehrere Formen der Datenübertragung ab und ist somit für verschiedene Anforderungen geeignet.

Damit der Einsatz von NFC-Technologie auch in sicherheitskritischen Anwendungen möglich ist, wurde die Technologie, die in Smart Cards für die Sicherheitsvorkehrungen existiert, in NFC integriert (vgl. Abschnitt 2.1.2). Dazu gehören unter anderem Sicherheitsdienste wie Verschlüsselung. Konkrete sicherheitstechnische Gefahren und entsprechende Schutzmaßnahmen werden in der Arbeit „Security in NFC“ von Haselsteiner und Breitfuß [7] ausführlich behandelt .

2.3 Abgrenzung zu anderen Technologien

NFC unterscheidet sich von anderen Übertragungstechnologien, wie Bluetooth, durch einige wichtige Merkmale. Dazu gehört die sehr kurze Entfernung zwischen den beiden beteiligten Geräten, die circa 0 bis 10 cm, maximal jedoch 20 cm beträgt. Die Datenübertragungsrate bei der Near Field Communication ist mit derzeit bis zu 424 kbit/s im Vergleich zu anderen drahtlosen Übertragungstechniken ebenfalls recht gering (s. Abb. 3). Diese beiden Eigenschaften stehen im Einklang mit dem Verwendungszweck von NFC, nicht sehr große Datenmengen über eine kurze Entfernung zu übertragen.

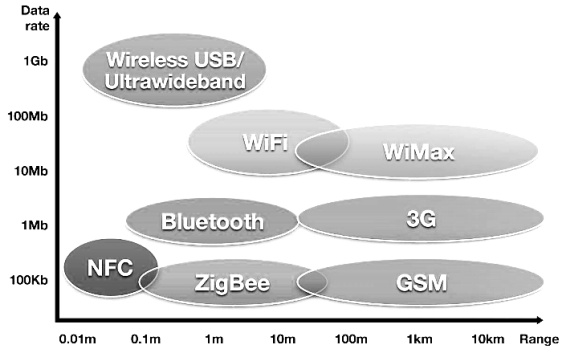


Abbildung 3: Drahtlose Verbindungstechniken [10]

Zudem zeichnet sich NFC durch den sehr schnellen Verbindungsaufbau aus, der innerhalb weniger Millisekunden stattfindet.
Im Vergleich zu Barcodes oder QR-Codes (Quick Response Codes) hat NFC den Vorteil, dass kein Sichtkontakt für die Datenübertragung notwendig ist. Abgesehen davon hat NFC umfangreichere und flexiblere Möglichkeiten, mit anderen Geräten oder Tags zu kommunziieren, wie beispielsweise die Peer-to-Peer-Kommunikation. Diese ist ein wichtiges Merkmal von NFC. Sie grenzt die Technologie auch von RFID ab, da RFID keine Peer-to-Peer-Kommunikation unterstützt.

3. ANWENDUNGEN

Langfristig könnte in vielen Gegenständen des Alltags NFC-Technologie integriert sein, wie in Mobiltelefonen, Türen, Fahrkartenautomaten, Parkuhren, Verkaufskassen und Geldautomaten. Vor allem der Einsatz in Mobiltelefonen ermöglicht viele neue Anwendungen. Im folgenden Abschnitt wird zunächst ein Einblick in die Einsatzbereiche gegeben. Anschließend werden einige konkrete Beispiele näher dargestellt.

3.1 Anwendungsbereiche

Die Anwendungsbereiche Service Initiation, Peer-to-Peer und Payment and Ticketing umfassen sämtliche Anwendungen,

die die Near Field Communication ermöglicht. Die Verbindung zwischen diesen Bereichen stellen NFC-fähige Geräte dar, die in Anwendungen aus allen Bereichen eingesetzt werden können, wie beispielsweise NFC-fähgie Mobiltelefone.

3.1.1 Service Initiation

Der Anwendungsbereich Service Initiation umfasst alle Anwendungen, bei denen mittels NFC Daten zu einer Dienstleistung oder einem Produkt abgerufen werden, wie die Adresse einer Website, eine Telefonnummer oder andere zusätzliche Informationen. Ein häufig genanntes Beispiel in diesem Zusammenhang stellen Smart Posters dar. Diese Plakate sind mit einem NFC-Tag ausgestattet und liefern weitere Informationen an ein NFC-Gerät, wenn es sich in einem entsprechenden Abstand befindet.

3.1.2 Peer-to-Peer

Beim Anwendungsbereich Peer-to-Peer werden zwischen zwei NFC-fähigen Geräten Daten ausgetauscht. Eine denkbare Anwendung ist hier die Verbindung zwischen einer Kamera und einem Drucker oder einem externen Bildschirm, um Fotos drahtlos auszudrucken beziehungsweise an einem Bildschirm zu betrachten. Des Weiteren ist die Datenübermittlung zwischen zwei NFC-fähigen Mobiltelefonen ein Beispiel für Peer-to-Peer-Anwendungen. Der Austausch der elektronischen Visitenkarten findet so durch Berührung der beiden Handys statt. Die Verbindung wird dabei automatisch initialisiert.

3.1.3 Payment and Ticketing

Payment and Ticketing umfasst sämtliche Anwendungen, in denen mittels NFC-unterstützter Geräte Bezahlungen erfolgen oder Zugangsberechtigungen in dem Gerät gespeichert werden und abrufbar sind. Zutrittssysteme mit NFC-Technik existieren bereits, ebenso wie elektronische Tickets. Auch die Möglichkeit mit NFC-fähigen Handys zu bezahlen, wurde vereinzelt schon umgesetzt.

3.2 Konkrete Anwendungsbeispiele

Im Folgenden werden einige konkrete Beispiele für Anwendungen mit NFC-Support beschrieben.

3.2.1 NFC im Mobiltelefon

Derzeit beträgt der Anteil an NFC-fähigen Mobiltelefonen circa 2 bis 3 Prozent. Die Verbreitung nimmt jedoch zu (vgl. Abb.4). Ein wesentlicher Anteil von NFC-fähgien Mobiltelefonen stellt eine Grundlage dar, um größere Projekte und Anwendungen zu verwirklichen und an den Kunden heranzutragen.
Die Abbildung 5 zeigt die Veranschaulichung der Komponenten in einem Mobiltelefon mit NFC-Support. Zusätzlich zu den normalen Funktionen eines Mobiltelefons, wofür der Application Processor und die SIM Card zuständig sind, gibt es in NFC-Handys weitere Anforderungen. Für die Kommunikation mit anderen NFC-Geräten, Tags und Lesegeräten existiert ein NFC-Controller. Dabei wird je nach Kommunikationsform und -partner über das Peer-to-Peer-Interface,

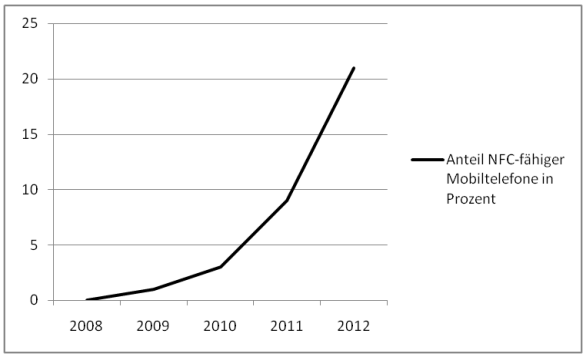


Abbildung 4: Anteil der NFC-fähigen Mobiltelfone am Gesamtmarkt [1]

das Reader-/Writer-Interface oder das Card Emulation Interface gearbeitet.
Für Anwendungen, die eine höhere Sicherheit erfordern, ist in diesem Modell von einem NFC-fähigen Mobiltelefon eine Smart Card integriert.
Die Kommunikation zwischen dem Application Processor

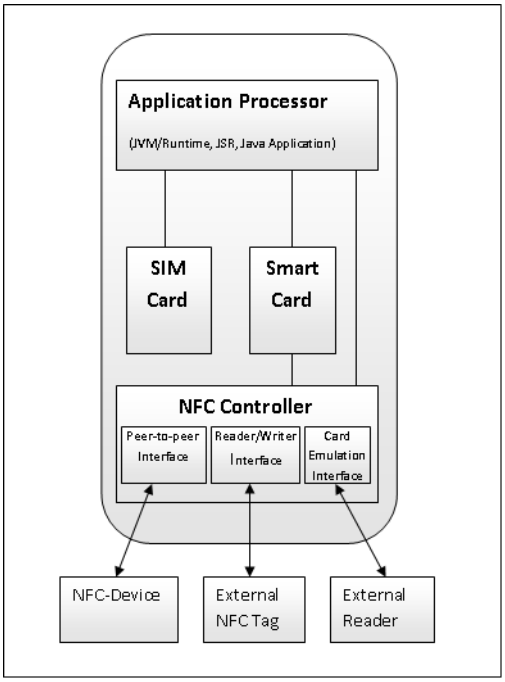


Abbildung 5: Komponenten in einem NFC-fähigen Mobiltelefon

und der SIM beziehungsweise der Smart Card erfolgt über so genannte Application Protocol Data Units (APDU). Diese sind Kommunikationseinheiten, die zwischen der Karte und der Anwendung übermittelt werden. Man unterscheidet dabei zwischen Kommando- und Antwort-APDUs [21]. Für eine direkte Kommunikation zwischen SIM Card und NFC-Controller kann das Single Wire Protocol (SWP) verwendet werden [9].

3.2.2 Touch&Travel

Durch NFC soll das Mobiltelefon zur mobilen Fahrkarte für öffentliche Verkehrsmittel werden. Die Testphase von Touch&Travel begann im Jahr 2008 und wurde seither auf mehrere Pilotgebiete erweitert, darunter Berlin und das Ruhrgebiet [2]. Alle Testkunden erhalten ein NFC-fähiges Mobiltelefon. Des weiteren existieren an jedem dazugehörigen Bahnhof sogenannte Touchpoints. In diesen ist ein passiver NFC-Tag integriert, dessen Daten von dem NFC-fähigen Handy gelesen werden können. Wenn Touch&Travel für eine Reise genutzt wird, muss sich der Passagier mit seinem Mobiltelefon bei Fahrtbeginn und Fahrtende an einem Touchpoint an- beziehungsweise abmelden. Dabei arbeitet das Handy mit dem Read/Write-Modus, agiert also wie ein Lesegerät. Aus den erfassten Daten ermittelt ein Hintergrundsystem die gefahrene Strecke und die benutzten Verkehrsmittel, sowie den Fahrpreis. Die Daten sämtlicher Fahrten werden monatlich gesammelt, in einer Rechnung zusammengefasst und an den Kunden versandt.

3.2.3 Mobile Payment

Mobile Payment, kurz M-Payment, bezeichnet sämtliche Bezahlvorgänge, bei denen der Zahlungspflichtige mobile elektronische Kommunikationstechniken einsetzt.
Bei der Umsetzung von Mobile Payment muss unterschieden werden, welchem Zweck die Anwendung dienen soll. Zum einen gibt es Bezahlmöglichkeiten, die auf geringe Beträge und bestimmte Einsatzszenarien begrenzt sind, zum anderen solche, die in jeder Zahlungsituation eingesetzt werden können.

- Ersteres wurde bereits durch mobilkom austria in einem Feldversuch realisiert. Dabei ermöglicht ein NFC-fähiges Mobiltelefon, am Automaten Produkte zu erwerben. Sobald der Kunde das Handy an den Touchpoint hält, erscheinen auf dem Display die notwendigen Informationen, um den Kauf abzuwickeln. Auf diese Art und Weise können Kunden Fahr-, Park-, Lottoscheine und Snacks am Automaten kaufen. In einem ersten Schritt könnte sich NFC in Mobiltelefonen daher als Zahlungsmethode für kleinere Beträge, wie in diesem Beispiel, durchsetzen.

- Um NFC in jeder Zahlungssituation einsetzen zu können, ist eine wichtige Voraussetzung, dass die Transaktionen auf bestehender Infrastruktur geschehen können. In diesem Zusammenhang spielt der Card-Emulation-Modus von NFC-Geräten eine wichtige Rolle. Zudem muss die Koordinierung zwischen Banken und Mobilfunkanbietern geklärt sein, wenn NFC-fähige Handys wie Kreditkarten eingesetzt werden sollen.

Ein großer Vorteil von Mobile Payment im Zusammenhang mit NFC-fähigen Handys ist die verbesserte Kontrolle des Benutzers. Der Nutzer kann den aktuellen Kontostand jederzeit einsehen. Jeder Zahlungsvorgang wird protokolliert und der Kunde kann ihn schnell und problemlos an seinem Mobiltelefon überprüfen.
An diesem Punkt stellt sich jedoch die Frage nach dem Datenschutz und wie Situationen gehandhabt werden, in denen ein solches Mobiltelefon verloren geht oder in falsche Hände gerät.

4. ZUSAMMENFASSUNG UND AUSBLICK

Die Near Field Communication stellt sich bei näherer Betrachtung als sehr vielseitige Technologie dar. Sie bietet verschiedene Kommunikationsformen an und die Datenübertragung selbst kann dabei ebenfalls auf unterschiedliche Art geschehen. Des Weiteren gibt es neben NFC-Geräten, die zwischen diesen Kommunikations- und Übertragungsformen wechseln können, auch einfache Tags, die nur gelesen werden können.
Die Handhabung der Datenübertragung mittels NFC gestaltet sich dabei unkompliziert und die Datenübertragung geht zügig vonstatten.
Durch die flexible Gestaltung der Technologie gibt es für NFC sehr viele Anwendungssmöglichkeiten. Als wichtigste Anwendung könnte sich die Integration von NFC-Technologie in Mobiltelefonen herausstellen. In Verbindung mit NFC-fähigen Handys ergeben sich weitere mögliche Anwendungen, bei denen ein Handy mit einem anderen NFC-Gerät oder Tag kommuniziert. Langfristig wird die Technologie daher wahrscheinlich den Trend verstärken, dass immer mehr Funktionen auf das Handy verlagert werden.
Der großflächige Einsatz von NFC-Anwendungen wird allerdings im Augenblick noch durch die mangelnde Verbreitung von NFC-fähigen Mobiltelefonen und anderen Geräten mit NFC-Support gebremst. Trotzdem lassen die vielen Feldversuche und Projekte, die sich mit NFC-Technologie beschäftigen, vermuten, dass sich immer mehr NFC-basierte Anwendungen im Markt durchsetzen werden.

5. LITERATUR

[1] ABI Research, Zugriff unter *Waving at the Future with NFC*, `http://www.nttdocomo.com/features/nfc/`, 2009

[2] Deutsche Bahn, *Touch&Travel*, `www.touchandtravel.de`, 2010

[3] T. Eichstädt, *Vortrag zur Studienarbeit - Entwicklung eines HITAG 1 Kartenlesers*, `http://www.rrzn.uni-hannover.de/fileadmin/ful/kolloquium/vortrag-eichstaedt_01.pdf`, Universität Hannover, 2005

[4] C. Enrique Ortiz, *An Introduction to Near-Field Communication and the Contactless Communication API*, `http://java.sun.com/developer/technicalArticles/javame/nfc/`, 2008

[5] K. Finkenzeller, *RFID-Handbuch*, Carl Hanser Verlag, ISBN: 3-446-22071-2, 2002

[6] O. Grimm, *RFID-Technologie - Aufbau, Funktionsweise und technische Anwendungen*, `http://www.enduroteam.de/olivergrimm/RFID-Technologie_Aufbau_Funktionsweise_und_technische_Anwendungen.pdf`, Technische Universität Ilmenau, 2004

[7] E. Haselsteiner, K. Breitfuß, *Security in Near Field Communication (NFC)*, Strengths and Weaknesses, `http://events.iaik.tugraz.at/RFIDSec06/Program/papers/002%20-%20Security%20in%20NFC.pdf`, Philips

Semiconductors

[8] Innovision Research & Technology, *Near Field Communication in the real world - Turning the NFC promise into profitable, everyday applications*, http://www.nfc-forum.org/resources/white_papers/Innovision_whitePaper1.pdf, 2006

[9] ISO/IEC JTC 1 Information Technology, *SC 17 Proposal for a new work item for a new part of ISO/IEC 7816 – Full-Duplex Single Wire Protocol for Smart Cards*, http://isotc.iso.org/livelink/livelink/fetch/-8913189/8913214/8913250/8913253/JTC001-N-8018.pdf?nodeid=4745751&vernum=-2

[10] J.Main, *NFC Technology Overview* http://www.nfc-forum.org/events/oulu_spotlight/Technical_Architecture.pdf, NFC-Forum, 2009

[11] C. Miu, *Sicherheit bei Smartcards*, http://www.informatik.uni-hamburg.de/WSV/teaching/sonstiges/EwA-Folien/Miu-Paper.pdf

[12] Konferenz Mobile Commerce Technologien und Anwendungen (MCTA), *M-Payment und NFC - Deutschland im internationalen Vergleich auf der Konferenz MCTA 2009 am 26.-27.1. in Berlin*, http://www.openpr.de/pdf/271286/M-Payment-und-NFC-Deutschland-im-internationalen-Vergleich-auf-der-Konferenz-MCTA-2009-am-26-27-1-in-Berlin.pdf, 2009

[13] mobilkom austria, *NFC: Die kontaktlose Schnittstelle*, www.mobilkom.at/de/nfc

[14] Mücke Sturm Company, *Chancen für mobile NFC-Services -Bestandsaufnahme und Ausblick*, http://www.ecc-handel.de/download/67687901/NFC+Studie+V+1+0+-+Short+Version.pdf, 2009

[15] NFC-Forum, *Essentials for Successful NFC Mobile Ecosystems*, http://www.nfc-forum.org/resources/white_papers/NFC_Forum_Mobile_NFC_Ecosystem_White_Paper.pdf, 2008

[16] NFC-Forum, *Near Field Communication in the real world – part 2 - Using the right NFC tag type for the right NFC application*, http://www.nfc-forum.org/resources/white_papers/Innovision_whitePaper2.pdf,

[17] NFC-Forum, *NFC-Forum - Frequently Asked Questions*, http://www.nfc-forum.org/resources/faqs/Frequently_Asked_Questions_About_NFC_Jan_2010.pdf, 2010

[18] NFCNews, *NFC Forum releases two new specifications*, http://www.nfcnews.com/2010/01/27/nfc-forum-releases-two-new-specifications, 2010

[19] A. Paus, *Near Field Communication (NFC) in Cell Phones*, http://www.crypto.rub.de/imperia/md/content/seminare/itsss07/slides_near_field_communication.pdf, 2007

[20] F. Resatsch, *Developing and Evaluating Ubiquitous Computing Applications Using The Example of Near Field Communication in Germany*, Technische Universität München, 2009

[21] M. Sievänen, *Application Protocol Data Unit*, http://www.tml.tkk.fi/Studies/T-110.497/2003/lecture4.pdf

[22] smartNFC, *Was ist NFC?*, http://smartnfc.com/index.php?option=com_docman&task=doc_view&gid=23&Itemid=13&lang=de, 2006

[23] VDI Nachrichten, *Handys öffnen Tür und Tor*, http://www.vdi-nachrichten.com/vdi-nachrichten/aktuelle_ausgabe/akt_ausg_detail.asp?cat=2&id=25488, 2005

[24] M. Ward, R. van Kranenburg, G. Backhouse, *RFID: Frequency, standards, adoption and innovation* , http://www.rfidconsultation.eu/docs/ficheiros/TSW0602.pdf, 2006

[25] M. Welzel, *Was ist und wie funktioniert NFC?*, http://smartnfc.com/index.php?option=com_docman&task=doc_view&gid=54&Itemid=13%E2%8C%A9=en, 2007

WiMAX

Matthias Eckert
Betreuer: Tobias Bandh
Seminar Future Internet SS2010
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: eckertm@in.tum.de

KURZFASSUNG

Immer mehr digitale Dienste, ob stationär oder mobil, erfordern heutzutage eine schnelle Datenverbindung. Besonders entlegene Regionen leiden unter einer Unterversorgung aufgrund des fehlenden Netzausbaus. Um diesen Teil der Bevölkerung mit Breitband-Internet zu versorgen und schnelle mobile Datenversorgung bieten zu können, werden Lösungsansätze gesucht. Als auf dem IEEE 802.16 Standard aufbauende Funktechnologie für breitbandige Datenübertragung über weite Distanzen stellt WiMAX (Worldwide Interoperability for Microwave Access) eine scheinbar geeignete Lösung dar. In Konkurrenz zu anderen, teilweise ähnlichen Technologien, steht WiMAX vor Problemen, die in diesem Paper diskutiert werden sollen.

Schlüsselworte

WiMAX, IEEE 802.16, Breitband-Internet, 4G, LTE

1. EINLEITUNG

Worldwide Interoperability for Microwave Access (WiMAX) ist ein auf dem IEEE 802.16 aufbauender Standard für schnelle Datenübertragungen über weite Strecken. Durch diese Eigenschaft ist der WiMAX Standard insbesondere für regionale Netzwerke angedacht. Dies hat auch die Bundesregierung erkannt und verspricht durch eine geeignete Technologie Mischung ein deutschlandweit, flächendeckendes Breitbandnetz für schnellen Internetzugang bis zum Ende des Jahres 2010 [3]. Eine Möglichkeit stellt hierbei die Verbreitung von Breitband-Internet mit Hilfe von Funktechnologien dar. Somit könnte in dieser Mischung auch WiMAX zum Einsatz kommen, mit dessen Hilfe regionale Netzwerke in schwach besiedelten Regionen errichtet werden sollen. Diese Netzwerke können Breitband-Internet sowohl in Regionen und Haushalte, die momentan vom Netz der Versorgungsgesellschaften nicht erfasst werden, als auch auf mobile Endgeräte bringen.

2. WIMAX

WiMAX basiert auf dem IEEE 802.16 [20], ein parallel zum IEEE 802.11 (Wireless LAN, Ethernet) entwickelter Standard, für drahtlose Metropolitan Area Networks (MAN). Es existieren mehrere Ausprägungen des IEEE 802.16 Standards. Tabelle 1 zeigt die aktuell bedeutsamsten Revisionen.

In diesem Paper sollen die beiden wichtigsten, fixed WiMAX und mobile WiMAX, betrachten werden. Ursprünglich waren Frequenzbänder im Bereich zwischen 10 und 66 GHz für den IEEE 802.16 vorgesehen. Dieser Frequenzbereich eignet

Bezeichnung	Definition
IEEE 802.16d-2004	fixed WiMAX
IEEE 802.16e-2005	mobile WiMAX
IEEE 802.16j-2009	Weiterentwicklung/Korrektur von mobile WiMAX

Tabelle 1: IEEE 802.16 Standards

sich besonders für Richtfunk. Da Richtfunk jedoch einen direkten Sichtkontakt zwischen Sender und Empfänger – die sogenannte Line of Sight (LoS) – vorrausetzt, wurde das Frequenzspektrum des Standards mit der Revision d auf 2 bis 11 GHz verschoben. Hiervon sind für WiMAX vor allem drei Frequenzbereiche vorgesehen: 2,5 GHz, 3,5 GHz und 5,8 GHz. In diesem Frequenzbereich bietet WiMAX eine theoretische Übertragungsreichweite von 50 Kilometern.

Frequenzbereich	Untergrenze	Obergrenze
2,5 GHz	2500 MHz	2690 MHz
3,5 GHz	3400 MHz	3600 MHz
5,8 GHz	5725 MHz	5850 MHz

Tabelle 2: vorgesehene Frequenzbereiche

3. IEEE 802.16

Standardisiert wurde der IEEE 802.16 im Dezember 2001. Als Standard für drahtlose MAN umfasste er ein Frequenzband von 10 bis 66 GHz mit Bandbreiten von 20, 25 und 28 MHz. Zwar ist eine Reichweite von bis zu 75 Kilometern gegeben, jedoch setzt der Standard eine direkte Sichtverbindung zwischen der Basisstation und der Empfangsantenne voraus. Dies wiederum hat eine komplexe Installation fest installierter Außenantennen in ausreichender Höhe zur Folge. Die maximale Datenrate liegt bei etwa 134 MBit/s [7].

3.1 IEEE 802.16d – fixed WiMAX

Im Januar 2003 und Juli 2004 wurden die Revisionen a und d vom IEEE verabschiedet. Die beiden Standards wurden letztendlich unter dem Synonym IEEE 802.16e-2004 vereint. Die Revision d spezifiziert heute den eigentlichen fixed WiMAX Standard. Ein möglicher Bereich für die Frequenzbänder wurde auf 2 GHz erweitert und gleichzeitig auf 11 GHz nach oben begrenzt. Die Bandbreite wurde als frei skalierbar im einem Bereich zwischen 1,5 und 20 MHz festgelegt. Für einen 20 MHz-Kanal liegt die maximale Datenrate für diesen Standard bei 75 MBit/s. Diese Entscheidung

senkte die Reichweite auf fünf, beispielsweise für WiMAX Router mit integrierter Antenne, bzw. 15 Kilometer für Anlagen mit fest installierter Außenantenne. Laborwerte attestieren weiterhin sogar eine theoretische Reichweite von 50 Kilometern [7].

3.2 IEEE 802.16e – mobile WiMAX

Die beiden Adjektive "fixed" (IEEE 802.16d) und "mobile" (IEEE 802.16e) wurden mit Einführung der Revision e im Dezember 2005 geprägt. 802.16-2005 wird daher auch als Synonym für die Revision e verwendet. In der neuen Revision wurden die Frequenzbänder in Frequenzbereiche zwischen 0,7 und 6 GHz verschoben, was eine typische Reichweite von 1,5 Kilometer zur Folge hatte. Allerdings ermöglicht es dieser Standard erstmalig einen Funkzellenwechsel im laufenden Betrieb vorzunehmen und somit dem Empfänger Mobilität zu bieten. Generell wurde bei diesem Standard besonderer Wert auf Mobilität gelegt. So wurde die Revision e auch mit dem Hauptaugenmerk auf Ressourcen schonende Verfahren entwickelt. Dem neuen Standard war es auch zu verdanken, dass der Zwang einer Sichtverbindung nun entfiel. Die neue Möglichkeit einer non-line-of-sight (NLOS) brachte jedoch anderweitige Probleme hervor, auf die in Abschnitt 4 weiter eingegangen wird. Die maximale Datenrate liegt bei 15 MBit/s [7].

4. MODULATIONSVERFAHREN

Durch die fehlende Richtcharakteristik der Antennen, in mobilen Endgeräten und den Wegfall der LOS sinkt die Chance auf eine ideale Empfangsqualität des Signals. So können Mehrfachreflexionen des Signals und der Umgang mit diesen problematisch sein. Auch der Aufenthalt innerhalb geschlossener Räume stellt die Technologie vor Probleme. Abhilfe schafft hierbei der Einsatz der, je nach Situation, geeigneten Modulationsverfahren.
Laut [19] gilt: Die Modulation beschreibt die Veränderung von Signalparametern eines Trägersignals in Abhängigkeit von einem modulierenden Signal. Hierbei werden die Signale miteinander multipliziert, d.h. moduliert. Auf der Seite des Senders befindet sich ein Modulator welcher die Nachricht auf das Trägersignal prägt, bevor die Nachricht über einen Kanal versendet und empfangsseitig durch einen Demodulator wieder zurückgewonnen wird. Der WiMAX-Standard nutzt für sich die im Folgenden erklärten digitalen Modulationsverfahren.

4.1 QPSK

Mit QPSK (quadrature phase shift keying) bezeichnet man ein Modulationsverfahren, mit dessen Hilfe zwei Bits auf ein Symbol codiert werden können. Ein Symbol bezeichnet dabei einen Übertragungsschritt. Dieses Verfahren wird bereits in anderen Mobilkommunikation-Standards, wie Universal Mobile Telecommunications System (UMTS) verwendet. Hierbei erfolgt eine Unterteilung des Signals in vier Phasenlagen zu 45, 135, 225 und 315 Grad. Jede Phasenlage repräsentiert dabei einen Zustand. Mit vier verschiedenen Zuständen erlaubt die Quadratur-Phasenumtastung – so der deutsche Begriff – zwei Bits pro Symbol zu übermitteln. QPSK ist dem binary phase shift keying (BPSK) entsprungen. BPSK stellt die niedrigste Form der Modulationsverfahren dar. Das Verfahren überträgt pro Symbol genau ein Bit. BSPK kommt auch bei WiMAX regelmäßig zum Einsatz. Aufgrund der geringen Datenübermittlung pro Symbol, ist BSPK das robusteste Modulationsverfahren. Es wird somit stets zu Beginn einer Übertragung eingesetzt um den Teilnehmern das nachfolgend eingesetzte Verfahren mitzuteilen. Anschließend wird die bestmöglichste Modulationsstufe zur eigentlichen Übertragung verwendet, um die zur Verfügung stehende Bandbreite optimal auszunutzen [11, 17].

4.2 QAM

Ein weiteres von WiMAX genutztes Verfahren ist die Quadraturamplitudenmodulation (QAM). Es stellt eine Erweiterung von QPSK dar. Dieses Modulationsverfahren ermöglicht jedoch eine nochmals erhöhte Übertragungsdichte, da es zwei Trägerfrequenzen simuliert. Diese beiden Trägerfrequenzen werden erzeugt indem die ursprüngliche Trägerfrequenz zusätzlich um 90 Grad – entspricht 1/4 der Wellenlänge – phasenverschoben wird. Die beiden Trägersignale werden mit zwei oder mehreren unterschiedlichen Signalen amplitudenmoduliert und anschließend zusammengeführt, indem sie addiert werden. Auch dieses Verfahren findet in bereits eingesetzten Standards, wie High Speed Packet Access (HSPA) Verwendung. WiMAX nutzt sowohl QAM16 als auch QAM64, wobei die Zahl die größtmögliche Anzahl der einnehmbaren Zustände angibt. Mit Erhöhung der Modulationstufe lässt sich zwar die Datenrate bei gegebener Bandbreite steigern, jedoch geht dies zu Lasten der Robustheit gegenüber Interferenzen. Sollte eine höhere Modulation vorgenommen werden, so müssen auch geeignete Übertragungsbedingungen gegeben sein, da die Dekodierung bei Interferenzen sonst ungleich schwieriger wird. Wie Abbildung 1 zeigt stellen Ungenauigkeiten bei der Übertragung, bei höheren Modulationsstufen ein Problem dar. Bei zu großer Streuung lässt sich das Signal nicht mehr genau einer Bit-Gruppe zu ordnen. WiMAX ist daher in der Lage, in Abhängigkeit von der Signalqualität, zwischen den verschiedenen Verfahren zu wechseln [12, 15].

5. OFDM/OFDMA

Das von WiMAX genutzte Verfahren OFDM (Orthogonal Frequency Division Multiplexing) findet hauptsächlich in Hochgeschwindigkeits-Mobilfunknetzen Verwendung, so beispielsweise auch in LTE (siehe: Abschnitt 9.3). OFDM bezeichnet ein Zugangsverfahren, welches die Übertragung eines breitbandigen Signals auf mehrere schmalbandige orthogonale Signale gleichzeitig ermöglicht. Diese schmalbandigeren Signale werden als Subträger bezeichnet. Einzelne oder mehrere dieser Subträger können unterschiedlichen Nutzern zugewiesen werden, um somit eine optimale Ausnutzung der Bandbreite zu erreichen [10]. Die Breite eines jeden Subträgers ist festgesetzt. Für mobile WiMAX liegt diese bei 10,94 kHz. Die genaue Anzahl der verwendeten Subträger ist jedoch abhängig von der verfügbaren Bandbreite. Steht mehr Bandbreite zur Verfügung, so werden mehr Subträger eingesetzt [18]. Dies gilt jedoch nur für mobile WiMAX. Der große Unterschied zwischen fixed und mobile WiMAX liegt in der Tatsache, dass fixed WiMAX unabhängig von der Bandbreite stets 256 Subträger verwendet. Aus diesem Grund wird dem OFDM-Verfahren im Zusammenhang mit mobile WiMAX gerne auch das Präfix "Scalable" vorgeschoben [9]. OFDM besitzt zusätzlich die Eigenschaft sehr robust gegenüber Signalverfälschungen zu sein. Interessant wird diese Eigenschaft für den Downlink

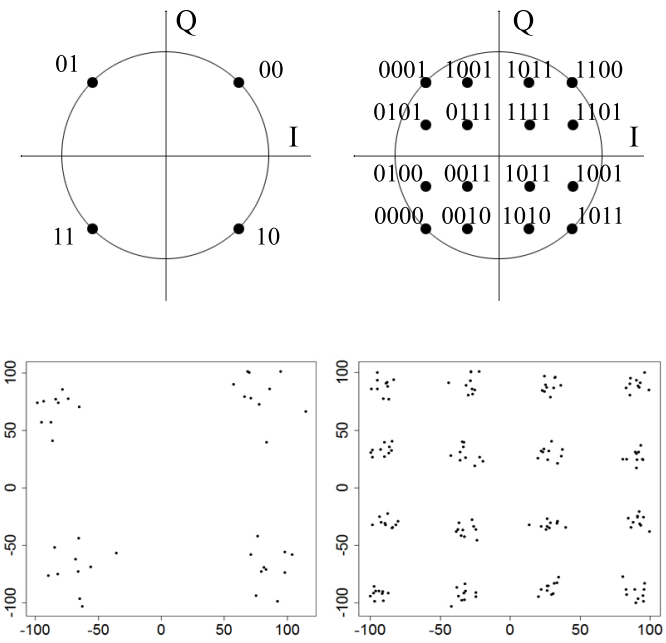


Abbildung 1: QPSK/QAM16 mit zugehörigen Messergebnissen

im Bandbreitenbereich über 5 MHz. Verfälschungen können empfangsseitig zwar auch eigenständig bereinigt werden, jedoch ist die hierfür benötigte Rechenkapazität unverhältnismäßig groß.
In der Praxis findet selten die Kommunikation zwischen einer Basistation und lediglich einem Empfänger statt. Meist sind an der Kommunikation mehrere Empfänger beteiligt. OFDM ist in der Lage auch mehrere Empfänger, so etwa die verschiedenen Mobiltelefone in einer Funkzelle, parallel zu bedienen bzw. von ihnen zu empfangen. Dies bezeichnet man als Nutzermultiplexing. Hierbei findet eine zeitlich festgelegte Übertragung auf unterschiedlichen Frequenzen, durch verschiedene mobile Endgeräte an die Basisstation statt.
In Richtung des Uplink wird OFDM daher gerne der Namen Orthogonal Frequency Division Multiplex Access (OFDMA) gegeben. Kritisch für OFDMA ist die unterschiedliche Entfernung zwischen den verschiedenen Mobilfunkgeräten und der Basisstation. Größere Entfernungen bedeuten größere Zeitabstände zwischen dem Absenden und Empfangen eines Signals. OFDMA ist allerdings auf das nahezu zeitgleiche Eintreffen der verschiedenen Subträger angewiesen. Die Basisstation terminiert, je nach Entfernung der Sender, den Versand daher zeitversetzt. Ein permanenter Zeitabgleich zwischen Basisstation und den mobilen Endgeräten ist daher unerlässlich. Alle Sender werden so gesteuert, dass alle ankommenden Signale innerhalb eines Sicherheitsintervalls (guard interval) liegen. Diesen aktiven Prozess der Abstimmung durch die Basisstation nennt man transmit-timing-control[15].

6. MIMO

Um die, teilweise komplexen, Modulationsverfahren optimal zu unterstützen ist ein möglichst guter Empfang der gesendeten Signale notwendig. Diese Unterstüzung leistet MIMO. Multiple Input Multiple Output (MIMO) bezeichnet ein Mehrfach-Antennen-System. Dabei werden senderwie empfängerseitig mehrere Antennen, mit dem Ziel ein qualitativ möglichst hochwertiges Empfangssignal zu erzielen, eingesetzt. Die entsprechende Anzahl an Antennen sollte jedoch auf beiden Seiten gleich sein. Entsprechend heißen die Verfahren 2 x 2 MIMO oder 4 x 4 MIMO. Die verschiedenen Antennen müssen mindestens den Abstand einer halben Wellenlänge der Trägerfrequenz zueinander besitzen. Diese Eigenschaft garantiert unterschiedliche Verbreitungswellen des Signals. Systeme in welchen MIMO zum Einsatz kommt, werden in mehrfacher Weise befähigt.
Zum einen lässt sich hierdurch die Versorgungsqualität steigern. MIMO Systeme haben die Auswahl zwischen mehreren Antennen, an welchen das Signal eingeht und können so je das qualitativ hochwertigste verwenden. Interferenzen werden durch das multiple Signal gesenkt und somit können beispielsweise der Zellenradius erweitert oder das Modulationsverfahren angehoben werden.
Zum anderen sind MIMO Systeme durch den Besitz mehrerer Antennen in der Lage parallel zu senden und zu empfangen. Alternativ lässt sich auch ein Verfahren namens "Spatial Multiplexing" nutzen. Mit Spatial Multiplexing werden mehrere Datenströme zeitgleich versandt. Sie benutzen dabei den selben Kanal (Frequenz), werden aber über unterschiedliche Antennen verbreitet. Der Empfänger kann dadurch räumliche Informationen gewinnen und besitzt zudem die Möglichkeit Mehrfachreflexionen kompensieren zu können. Der Nachteil der fehlenden Sichtverbindung wird hierbei aufgehoben. Auftretende Mehrfachreflexion innerhalb von Gebäuden können somit genutzt werden.
Positiv wirkt sich dies auf die Übertragungsrate aus, welche in Abhängigkeit der Anzahl verwendeten Sende- und Empfangsantenne – unter optimalen Bedingungen – verdoppelt werden kann [16, 18].

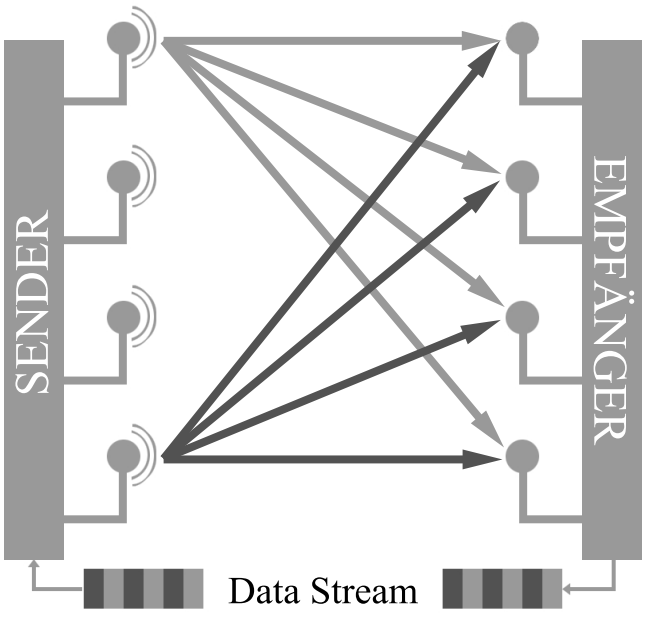


Abbildung 2: 4x4 MIMO mit Spatial Multiplexing

7. DUPLEXING

Unter dem Begriff Duplexing, auch Multiplexverfahren genannt, vereint man Verfahren zur Regelung des Verhaltens des Up- und Downlinks drahtloser Zwei-Wege-Kommunikation. Die zwei Verfahren, die es hier zu nennen gilt, sind: Frequency Divison Duplex (FDD) – hierbei ist wiederum zwischen full duplex FDD und half duplex FDD zu unterscheiden – und Time Division Duplex (TDD). Relevant für die Unterscheidung der beiden Verfahren ist vor allem das Frequenzspektrum auf dem sie arbeiten. Sowohl full duplex FDD, als auch half duplex FDD arbeiten auf einem "gepaarten Spektrum". Beide Verfahren nutzen somit zwei separate Frequenzbereiche für Down- und Uplink. Während jedoch full duplex FDD parallel auf beiden Frequenzen arbeitet, teilt half duplex FDD den beiden Signalen nicht überlappende Zeit-Slots zu, in welchen gesendet bzw. empfangen wird.

Ebenso wie half duplex FDD, teilt TDD für den Up- und Downlink verschiedene Zeit-Slots zu, innerhalb welcher gesendet bzw. empfangen wird. Der Unterschied liegt darin, dass TDD einen gemeinsamen Frequenzbereich für Up- wie Downlink nutzt. TDD verwendet somit ein "ungepaartes Spektrum". Problematisch ist hierbei die Tatsache, dass die Höchstgeschwindigkeit für Up- und Downlink im Gegensatz zu FDD nie gleichzeitig erreicht werden kann [15].

Die WiMAX Spezifikation [20] sieht den Einsatz beider Duplexing-Verfahren vor. Es hat sich jedoch in der Praxis gezeigt, dass TDD deutlich populärer unter den Providern ist [13].

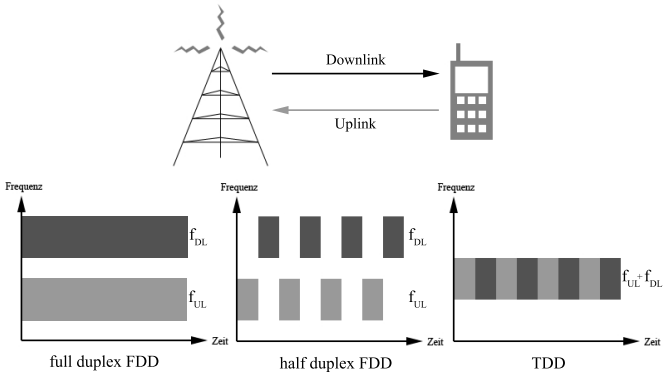


Abbildung 3: frequency- and time-division duplex

8. FREQUENZBEREICHE

Einen Vorteil seiner Technik trägt WiMAX bereits im Namen: Worldwide. Durch die Nutzung der Frequenzbereiche zwischen 2 und 11 GHz ist WiMAX prinzipiell weltweit nutzbar. Problematisch ist jedoch die eventuell vorhandene staatliche Regulierung und ein damit verbundenes Frequenz-Zuteilungsverfahren in den unterschiedlichen Regionen der Welt.

In Deutschland wurden die Frequenzen im Bereich zwischen 3,4 GHz und 3,6 GHz aufgrund des zunächst starken Interesses mit zehnmonatiger Verspätung im Dezember 2006 durch die Bundesnetzagentur für 56 Millionen Euro versteigert [4]. Dies stellt jedoch nur einen Bruchteil der Summe, die durch die Vergabe der UMTS-Lizenzen im Jahre 2000 erzielt wurde, dar. Mit der Ersteigerung der neuen Frequenzen gingen die drei Lizenznehmer [7] die Verpflichtung

Region	Frequenzbereiche
Europa	3,5 GHz / 5,8 GHz
Russland	3,5 GHz / 5,8 GHz
Mittlere Osten	3,5 GHz / 5,8 GHz
Afrika	3,5 GHz / 5,8 GHz
Asien/Pazifik	2,3 GHz / 2,5 GHz / 3,3 GHz / 3,5 GHz / 5,8 GHz
Kanada	2,3 GHz / 2,5 GHz / 3,5 GHz / 5,8 GHz
USA	2,5 GHz / 5,8 GHz
Mittel-/Südamerika	2,5 GHz / 3,5 GHz / 5,8 GHz

Tabelle 3: Verfügbare Frequenzbereiche

Quelle:http://www.elektronik-kompendium.de/sites/net/0904211.htm

ein, bis Ende 2009 15 Prozent, bis Ende 2011 25 Prozent und bis 2014 sogar 75 Prozent der deutschen Haushalte mit Breitband-Internet zu versorgen [3, 4]. Der deutsche Frequenzbereich, für ein in der Zukunft mögliches Deployment von fixed WiMAX sollte damit festgelegt sein. Im Dezember 2009 kündigte die Bundesnetzagentur an, neben anderen, zusätzlich die Frequenzen im Bereich zwischen 790 MHz und 862 MHz versteigern zu wollen. Im Volksmund als "Digitale Dividende" bezeichnet, wurden diese Frequenzen im Zuge der Umstellung von analogem auf digitales Fernsehen wieder frei. Sie gelten als besonders geeignet für Übertragungen über weite Distanzen. Dies spiegelt auch die Tatsache wider, dass vor Ende der Auktion bereits im April 2010 abzusehen war, dass diese Frequenzbereiche den größten Teil der Auktionserlösen ausmachen werden. Die Bundesregierung erhofft sich durch die Versteigerung "die Versorgung der Bevölkerung mit breitbandigen Internetanschlüssen, insbesondere in ländlichen Gebieten" [5] beschleunigen zu können. Hierbei wird der Konflikt zwischen fixed und mobile WiMAX ersichtlich. War zunächst ausschliesslich fixed WiMAX für die Versorgung dieser ländlichen Gebiete angedacht, so dringt die Revision e des WiMAX Standards nun auch in dieses Einsatzgebiet vor.

9. DIE PRAXIS

Ursprünglich trat WiMAX an den Start um entlegene Regionen mit Breitband-Internetanschlüssen zu versorgen. Die Überbrückung "der letzten Meile" – in Deutschland ein Monopol der Deutschen Telekom, welches den Anschluss von Verbrauchern an das Festnetz bezeichnet – war dabei das ausgegebene Ziel. Einen Vorteil stellt hierbei die Möglichkeit, WiMAX Sendeantennen (Basisstationen) als Kette zu installieren und somit mehrere Kilometer überbrücken zu können, dar. Somit kann der typische Zellenradius, der normalerweise nur einige Kilometer beträgt, um ein Vielfaches vergrößert werden. Eine verhältnismäßig starke Lobby treibt ihre Forschung und damit verbundene Entwicklungen immer weiter voran. Intel beispielsweise produziert bereits Chipsätze für mobile Endgeräte und ermöglicht somit bereits Internet der vierten Generation (4G). Wohingegen bei dem großen Konkurrenz-Standard Long-Term-Evolution (LTE) erst zwischen 2010 und 2012 mit einer kommerziellen Nutzung zu rechnen ist [18]. Tatsächlich wurden mit Beginn des Jahres 2010 erste LTE-fähige Geräte auf den Markt

gebracht. Einige Jahre nach den ersten WiMAX-fähigen Endgeräten. Trotz des zeitlichen Vorsprungs und der starken Lobby hinter WiMAX, steht die Technologie in der Praxis vor einigen Problemen.

9.1 Ein Standard der mit sich selbst konkurriert

Zwar waren die beiden Standards IEEE 802.16d und IEEE 802.16e zunächst für unterschiedliche Szenarien gedacht, jedoch zeigt sich vermehrt ein nachvollziehbares Interesse die beiden Standards vereinen zu wollen. Dies gestaltet sich jedoch problematisch. Während bei mobile WiMAX die Subträgeranzahl mit der zur Verfügung stehenden Bandbreite variiert, nutzt fixed WiMAX stets 256 Subträger. Aufgrund der unterschiedlichen Zugangsverfahren derer sich die beiden Standards bedienen, sowie dem Zwang einer Sichtverbindung in der Revision d des WiMAX Standards, ist eine Kompatibilität von fixed und mobile WiMAX somit leider nicht gegeben. Auch der Handover einer Verbindung, eines sich bewegenden mobilen Endgerätes, zwischen zwei Funkzellen wird vom IEEE 802.16d Standard nicht unterstützt [18]. Für Provider fällt die Entscheidung zugunsten eines Standards schwer. Sie möchten möglichst beide Vorteile aus der Technologie mitnehmen: die Möglichkeit ihren Kunden WiMAX als DSL-Ersatz anbieten zu können, bei gleichzeitiger Bedienung des Marktes für mobile Endgeräte. Potenzielle Anbieter von WiMAX werden jedoch gezwungen, sich für einen Standard entscheiden zu müssen bzw. beide parallel anbieten zu müssen. Dies stellt leider einen großen Nachteil der Technologie dar.

9.2 WiMAX als Ersatztechnologie

Durch den mittlerweile bereits weit voran geschrittenen Ausbau Netzes der Deutschen Telekom, welcher auch weiterhin stark voran getrieben wird [2], ist diese in der Lage mehr und mehr Haushalte mit DSL zu versorgen. Es existieren momentan nur wenige kommerziell genutze WiMAX Netze in Deutschland. Diese werden größtenteils zusätzlich auch weiterhin zu Testzwecken genutzt. Durch die mittlerweile sehr ausgedehnte Time-to-Market stellt sich die Frage nach der Notwendigkeit von WiMAX als Ersatztechnologie somit immer mehr. Erschwerend für Lizenznehmer kommt die in Abschnitt 8 genannte Lizenzverpflichtung hinzu. Die Regierung wünscht sich möglichst viele regionale Haushalte an das Breitband-Internet anschließen zu können und legt daher den Fokus auf dünn-besiedelte Gebiete. Für WiMAX-Anbieter stellt dies ein Problem dar. Eine priorisierte Anbindung von kleinen Gemeinden bedeutet für sie in erster Linie, eine geringe Zahl an Kunden. Somit stellt sich für viele Unternehmen die Frage der Rentabilität eines solchen Projekts.
Gerade auch aufgrund der Tatsache, dass unter den Anbietern große Unklarheiten bezüglich der von ihnen zu praktizierenden Geschäftsmodelle herrscht. Nach dem sich einstellenden Misserfolg der WLAN-HotSpots herrscht große Verunsicherung. Der umständliche Anmeldeprozess und der damit verbundene Umstand sich mit einer Kennung nicht in beliebigen Netzwerken anmelden zu können, wurden von vielen potentiellen Kunden abgelehnt. Ein ähnliches Problem könnte WiMAX drohen.

9.3 LTE

Natürlich steht der WiMAX-Standard nicht alleine auf dem Spielfeld des Breitband-Internets per Funk. Neben ihm existiert ein weiterer Standard, der die selben Ziele wie auch WiMAX verfolgt: LTE – kurz für Long Term Evolution. LTE ist ein von dem 3rd Generation Partnership Project (3GPP) [1] voran getriebener Standard. Das 3GPP ist auch für die Standardisierung und Weiterentwicklung von GSM und seiner Nachfolgetechnologien verantwortlich. LTE wurde im Dezember 2008 festgesetzt [8].
LTE weist viele Gemeinsamkeiten mit WiMAX auf. So ist LTE ein Standard, der ebenfalls die Eigenschaft einer hohen Spitzen-Datenrate von bis zu 108 MBit/s bietet. Die hierfür verwendeten Zugangsverfahren OFDM und OFDMA (siehe Abschnitt 5) finden sich in beiden Standards wieder. Weiter verwenden beide Standards die Modulationsverfahren QPSK und QAM. Mit 1.4 bis 20 MHz weist LTE annähernd genau den gleichen Bandbreitenbereich wie WiMAX auf. Diese Ähnlichkeit besitzen auch die Frequenzbreiten der Subträger die bei WiMAX mit 10.94kHz und bei LTE mit 15 kHz angegeben werden [8].
Es existieren jedoch auch Unterschiede. So sieht beispielsweise LTE, aufgrund der begrenzten Antennengröße mobiler Geräte, für den Uplink kein MIMO-Verfahren vor. Während der Standard von WiMAX jedoch eine vereinfachte Form, die Single-User-MIMO (SU-MIMO) Technik spezifiziert, unterstützt LTE hingegen theoretisch die Weiterentwicklung Multi-User-MIMO (MU-MIMO), welche in der Lage ist, den oben genannten Nachteil auszugleichen. Der wichtigste Unterschied ist die unterschiedliche Verwendung von Multiplexverfahren. WiMAX unterstützt zwar sowohl FDD als auch TDD, jedoch findet in der Praxis bei WiMAX-Providern fast ausschließlich TDD Verwendung [6]. LTE hingegen bedient sich FDD.
Trotz der teilweise existierenden Überschneidungen in einigen Bereichen, reichen die Unterschiede der beiden Standards um WiMAX und LTE nicht zu einem gemeinsamen Standard vereinen zu können.
Da LTE ein Standard der 3GPP ist, besitzt er WiMAX gegenüber einen entscheidenden Vorteil: Als Standard, der einer Release-Reihe von Standards aufbauend auf GSM entspringt, ist LTE rückwärtskompatibel zu bereits implementierten Standards, wie z.B. HSPA. LTE sieht sowohl die Möglichkeit eines Deployments als Stand-Alone-Lösung vor, als auch – und dies ist mit Sicherheit für viele Anbieter die interessanter Variante – als in den GSM Standard integrierte Lösung. Ein Fallback auf HSPA oder geringer in "LTE-Funklöchern" stellt somit kein Problem dar. WiMAX sieht eine solche Möglichkeit aktuell nicht vor.

10. ZUSAMMENFASSUNG

Als neuer Standard mit hoher Datentransferrate, sowie einer hohen Reichweite sollte WiMAX ideal für heutige Anforderungen gewappnet sein. Nicht nur die Möglichkeit der Überbrückung "der letzten Meile" die WiMAX bietet, sondern auch ein großes Forum mit aktuell mehr als 360 Mitgliedern (Stand: April 2010) [14] stellen einen großen Vorteil des Standards dar. Die anfängliche Idee zwei verwandte Standards für zwei unterschiedliche Einsatzszenarien zu entwickeln, entpuppt sich jedoch mittlerweile als eine Art Blockade für den weltweit kommerziellen Einsatz von WiMAX. Die bereits existierenden Lösungen des IEEE 802.15d Standards ermöglichen mobilen Endgeräten keine Nutzung der vorhan-

	LTE	802.16d	802.16e
Zugangsverf.(UL)	OFDM	OFDM	OFDM
Zugangsverf.(DL)	OFDMA	OFDMA	SOFDMA
Sichtverbindung	NLoS	LoS	NLoS
Bandbreite (in MHz)	1.4, 3, 5, 10, 15, 20	1,5 - 20	1,5 - 20
Subträgerbreite	15 kHz	10,94 kHZ	10,94 kHZ
Subträgeranzahl	variabel	256	variabel
Modulation	QPSK, 16QAM, 64QAM	QPSK, 16QAM, 64QAM	QPSK, 16QAM, 64QAM
Duplexing	FDD	TDD, FDD	TDD, FDD

Tabelle 4: Vergleich WiMAX - LTE

denen Netze. Gerade Nutzer mobiler Endgeräte stellen für Provider jedoch ein weiter wachsendes Geschäftsfeld dar.
Unterschiedlich verfügbare bzw. genutzte Frequenzbereiche und damit verbundene fehlende internationale Kooperation zur Durchsetzung eines Standards, sind ein weiterer Grund für die Behinderung des WiMAX Standards. Eine fehlende internationale Marschrichtung der nationalen als auch internationalen Konkurrenz macht die Entscheidung zugunsten WiMAX für potentielle Provider um ein Vielfaches schwieriger. Nur wenn sie sicher sein können, dass die Konkurrenz auf die selbe Technologie setzt, rentieren sich Investitionen. Dies ist zum einen dadurch bedingt, dass eine zu geringe Marktdurchdringung des Standards Verbraucher verunsichern oder gar verschrecken könnte. Andererseits könnte die Refinanzierung der Investitionen der Provider gefährdet sein, sollten ihre Kunden nicht die Möglichkeit besitzen auch im Ausland mobile Datenübertragung per WiMAX nutzen zu können. Da Roaming-Gebühren eine wichtige Einnahmequelle für Mobilfunk-Provider darstellen, sind diese Einnahmen für die Refinanzierung unerlässlich.
Selbiges gilt auch für Produzenten von WiMAX-Hardware für Endgeräte, wie Intel. Diese sind bereits in der Lage WiMAX-fähige Hardware zu produzieren und wären somit theoretisch ebenso in der Lage die Nachfrage nach WiMAX künstlich zu steigern indem entsprechende Hardware verbaut wird. Das Risiko, herauszufinden ob Kunden bereit sind einen Mehrpreis für bislang "unnütze" Bauteile zu zahlen, wollen sie jedoch nicht tragen.
Den, durch eine früherer Festsetzung des Standards gegebenen zeitlichen Vorsprung gegenüber LTE, scheint WiMAX nicht optimal zu nutzen. Zwar gibt es im Gegensatz zu LTE bereits funktionstüchtige, kommerziell genutzte Netze, so beispielsweise Teile des Netzes des Mobilfunkanbieters Sprint in den USA [18], jedoch scheint sich aufgrund der Tatsache, dass LTE in einem früheren Teststadium bereits ähnlich gute bzw. bessere Ergebnisse im Bereich der Übertragungsgeschwindigkeiten erzielen konnte, der Markt in Richtung LTE zu kippen. Die Eigenschaft der Rückwärtskompatibilität von LTE tut ihr übriges. Seit jedoch der Netzwerkausrüster Cisco im März 2010 bekannt gab, sich aus der Entwicklung von WiMAX-Basisstationen zurückzuziehen, scheint die Entscheidung zugunsten LTE's gefallen zu sein.
Wie bei jedem Kampf zweier zukunftsträchtiger Standards, ist es eher unwahrscheinlich, dass beide in Koexistenz überleben werden. Eine Niederlage des WiMAX Standards ist nicht ausgeschlossen und wird durch neueste Entscheidungen des Marktes immer wahrscheinlicher.

11. LITERATUR

[1] About 3GPP. http://www.3gpp.org/About-3GPP.

[2] Breitband DSL: Telekom investiert 10 Milliarden Euro. http://www.areamobile.de/news/14623-breitband-dsl-telekom-investiert-10-milliarden-euro.

[3] Cebit: Deutschland setzt auf Breitband. http://www.bundesregierung.de/Content/DE/Artikel/2009/03/2009-03-02-bk-cebit-eroeffnung.html.

[4] Drei neue bundesweite Breitband-Dienstleister. http://bwa-versteigerung.bundesnetzagentur.de/images/pressemitteilungen/06-12-15

[5] Eckpunkte der Bundesnetzagentur für Elektrizität, Gas, Telekommunikation, Post und Eisenbahn für die Vergabe von Frequenzen im Bereich von 790 MHz bis 862 MHz für den drahtlosen Netzzugang zum Angebot von Telekommunikationsdiensten. http://www.bundesnetzagentur.de/media/archive/15746.pdf.

[6] FDD/TDD: WiMAX and LTE Crossing Paths? http://www.wimax.com/commentary/blog/blog-2009/october-2009/fdd-tdd-wimax-and-lte-crossing-paths-1015.

[7] IEEE 802.16 / WiMAX. http://www.elektronik-kompendium.de/sites/net/0904211.htm.

[8] LTE. http://www.3gpp.org/LTE.

[9] OFDM Parameters in WiMAX (Cont). http://www.wimax.com.

[10] OFDMA (orthogonal frequency division multiplexing access). http://www.itwissen.info/definition/lexikon/orthogonal-frequency-division-multiplexing-access-OFDMA.html.

[11] Quadratur-Phasenumtastung. http://www.itwissen.info/definition/lexikon/Quadratur-Phasenumtastung-QPSK-quadrature-phase-shift-keying.html.

[12] Quadraturamplitudenmodulation. http://www.itwissen.info/definition/lexikon/Quadraturamplitudenmodulation-QAM-quadrature-amplitude-modulation.html.

[13] What is IEEE 802.16d? http://www.wimax.com/education/faq/faq44.

[14] WiMAX Forum - Member Roster. http://www.wimaxforum.org/about/member-roster.

[15] E. Dahlmann, S. Parkvall, J. Sköld, and P. Beming. *3G Evolution – HSPA and LTE for Mobile Broadband*. Elsevier Ltd., Burlington, USA, 2008.

[16] M. Riegel, D. Kroeselberg, and A. Chindapol. *Deploying Mobile WiMAX*. Wiley, Chichester, UK, 2009.

[17] P. D.-I. D. Rudolph. Digitale und Analoge Modulationsverfahren. Technical report, Technische Fachhochschule Berlin, 2005/2006.

[18] M. Sauter. *Beyond 3G - Bringing networks, terminals and the web together*. Wiley, Chichester, UK, 2009.

[19] K. Steudler. Übertragungtechnik - Modulation. Technical report, Hochschule für Technik und Architektur Bern, 2003.

[20] The Institute of Electrical and Electronics Engineers, Inc., New York. *IEEE Standard for Local and metropolitan area networks - Part 16: Air Interface for Broadband Wireless Access Systems*, May 2009.

Network Resilience

Tsvetko Tsvetkov
Seminar Future Internet, SS 2010
Institut für Informatik, Lehrstuhl Netzarchitekturen und Netzdienste
Technische Universität München
tsvetko.tsvetkov@mytum.de

Kurzfassung

Dass Widerstandsfähigkeit des Internets ein aktuelles und wichtiges Thema ist, zeigen die wissenschaftlichen Diskussionen sowie die Forschungsprojekte auf nationaler und internationaler Ebene. Die Nutzung des Internets hat sich in den letzten vier Jahrzehnten sehr stark verändert. Die damaligen Entwickler konnten die rapide Weiterentwicklung nicht voraussehen und es ist nicht überraschend, dass viele an das Internet gestellte Anforderungen nicht mehr oder nur noch unzureichend erfüllt werden können. In diesem Paper werden verschiedene bekannte und zukünftige Technologien betrachtet, die viele Schwachstellen abdecken und somit die Netzwerk-Resilienz erhöhen.

Schlüsselworte

Future Internet, Netzwerk, Resilience, Fehlertoleranz, Robustheit, Sicherheit, Zuverlässigkeit

1. Einführung

Das Internet ist mittlerweile zu einem globalen und sozialen Phänomen geworden. Die Gesellschaft hängt für nahezu jeden Aspekt des täglichen Lebens von Netzwerken ab. Täglich werden Dienste wie World Wide Web, E-Mail, Dateiverwaltung, Chat verwendet und dabei ist es wichtig in nicht vorhersehbaren Situationen, die beispielsweise durch Störungen zwischen Switches/Routern oder komplett Ausfällen von Netzknoten entstehen, adäquat reagieren zu können. Aus Sicht eines Anbieters hängt die Stabilität des Dienstes von der Qualität und Absicherung des Servers ab. Hier sind neben der reinen Serverleistung Faktoren wie Standort, Sicherungsmaßnahmen und Notfallpläne (z.B. Protection Switching) wichtig.

Die Erhöhung der Widerstandsfähigkeit des Internets ist auch ein zentrales Thema für Organisationen wie die *IFIP Working Group* und der EU-Kommission durch das *ResumeNet* [1] Projekt geworden. Im Weiteren werden die Definition der Netzwerk-Resilienz (Abschnitt 2) und die Ursachen, warum das Internet nicht widerstandsfähig genug ist (Abschnitt 3), erwähnt. Zusätzlich werden im Abschnitt 4 bekannte Techniken zur Erhöhung der Resilienz betrachtet. Abschnitt 5 behandelt zukünftige Ansätze und die Ausarbeitung wird schließlich mit den Abschnitten 6 und 7 – Zusammenfassung und Literatur-referenzen abgeschlossen.

2. Definition

Der Begriff der Resilienz (engl. Resilience) wird in verschiedenen Bereichen der Wissenschaft verwendet. In der Psychologie wird beispielsweise die Resilienz als die Stärke eines Menschen bezeichnet, Lebenskrisen wie schwere Erkrankungen oder Unfälle zu überwinden.

Im Netzwerkbereich wird der Begriff der Resilienz als die Möglichkeit bezeichnet, ein hohes Leistungsniveau zu erbringen, auch wenn der Normalbetrieb durch unerwartete Faktoren gestört wird. Ein klassisches Beispiel für solche Faktoren sind u.a. Computerwürmer, die den IPv4 Bereich eines Subnetzes scannen um Sicherheitslücken bei den Endgeräten zu finden und somit Zielrechner infizieren zu können. Im Jahre 2003 wurde *W32.Blaster* [2] berühmt, der eine Sicherheitslücke in der RPC-Schnittstelle von Microsoft Windows ausgenutzt und sich somit verbreitet hat. Dabei sollte der Wurm einen gezielten DDoS-Angriff gegen die Updateseiten von Microsoft starten.

Ein weiteres Beispiel wären Topologieänderungen des Netzes. Speziell in geswitchten Umgebungen sollte man auf der Sicherungsschicht (ISO-OSI Modell) zu jedem möglichen Ziel immer nur einen Pfad haben, da ansonsten Datenpakete mehrmals eintreffen können und somit Fehlerfunktionen beim Empfänger auslösen können.

Man muss allerdings betonen, dass Resilienz als Obermenge von Begriffen, wie Überlebensfähigkeit (engl. survivability) und Verfügbarkeit (engl. availability), verwendet werden kann [3]. Dabei bezeichnet „availability“ die Möglichkeit ein bestimmtes System oder einen bestimmten Dienst zu verwenden. Ein 100 % überlebensfähiges System würde zusätzlich einen zeitgenauen Dienst anbieten können. Es würde theoretisch im Falle eines Angriffes oder Störung keine Zeit brauchen um einen „Notfallplan“ zu realisieren. Dabei werden keine Datenpakete, die in der „Planwechselphase“ eintreffen, verloren gehen.

3. Gründe für die nicht-Resilienz

Die Geschichte des Internets beginnt bereits vor 40 Jahren. Durch den Aufbau des ARPAnet, das den Zweck hatte, Universitäten und Forschungseinrichtungen zu vernetzen, wurden die Grundlagen erforscht und entwickelt. Damals wurde das *Network Control Protocol (NCP)* entwickelt, was Adressierung und Transport in sich kombiniert hat. Zusätzlich wurden die Grundbausteine der heutigen Protokolle, wie *File Transfer* und *Remote Login* gelegt. Nachdem man am 1 Januar 1983 [18] den großen Schritt gemacht hat und von NCP nach TCP/IP umgestiegen ist, wurde manchen klar, dass *Flag Days* negative Konsequenzen haben werden. Das Internet hatte damals ca. 400 Netzknoten und es war durchaus möglich einen planmäßigen Netzausfall durchzuführen. Solche Maßnahmen sind heutzutage undenkbar und erschweren somit den Entwurf von neuen Netz-Architekturen.

Allerdings darf man sich nicht darauf verlassen, dass neue Innovationen zu einem widerstandsfähigen und stabilen Internet führen werden. Obwohl TCP einen enormen Fortschritt geleistet hat, hatten die damaligen Netzwerke mit Netzüberlastungen zu kämpfen. Die Netz-Knoten waren komplett überlastet und man konnte keinen zuverlässigen Dienst anbieten. Die Lösung des Problems war die TCP Staukontrolle [23]. Dieser Schritt hat nicht nur einen Ausweg von den Stausituationen gefunden, sondern hat auch neue Hürden für alle zukünftigen Techniken gestellt: Rückwärtskompatibilität und inkrementelle Erweiterbarkeit.

Das Internet wurde so konstruiert, dass es auch nach einem Atomschlag funktionsfähig bleibt. Dass dieser Mythos jedoch nicht den Tatsachen entspricht, wurde bereits in den letzten Jahrzehnten gezeigt. Beispielsweise hatte der Stromausfall im US-Bundesstaat Virginia im Jahre 2001 sogar in Deutschland seine Auswirkungen. Dabei waren zahlreiche Web-Dienste bundesweit nicht erreichbar [22].

Im Folgenden werden weitere Probleme des heutigen Internets geschildert, die zur Störung der Funktionsfähigkeit führen können.

3.1 Routing

Routing wird in der Telekommunikation als das Festlegen von Wegen zur Vermittlung von Paketen in Rechnernetzen bezeichnet. Häufig wird Routing auch als die allgemeine Übermittlung von Nachrichtenpaketen betrachtet, da es meistens eine Zusammensetzung von den Begriffen *Wegbestimmung* und *Weiterleitung* (engl. Forwarding) ist. Der Begriff Forwarding beschreibt einen Entscheidungsprozess eines einzelnen Knotens. Dabei kann ein Datenpaket unter bestimmten Voraussetzungen verworfen werden. Beispielsweise könnte ein Switch das CRC-Verfahren verwenden, um fehlerhafte Pakete zu erkennen und somit deren Weiterleitung zu verbieten (z.B. Store-and-Forward Methode).

Die optimale Wegbestimmung mit Hilfe von Routing-Protokollen ist ein wesentliches Element des heutigen Internets geworden. Das bezieht sich nicht nur auf die Kommunikation innerhalb eines Internetproviders, sondern auch besonders auf den Datenverkehr zwischen *autonomen Systemen (AS)*. Aufgrund des exponentiellen Wachstums der Internet-Nutzung in der Periode zwischen 1980-2005 (Abbildung 1) können heute *Reaktionszeiten* von Routing-Protokollen zwischen Providern sogar Sekunden dauern. Zusätzlich besteht die Gefahr, dass *Fehlerkonfigurationen* von Protokollen schnell Netzwerk-Ausfällen verursachen können.

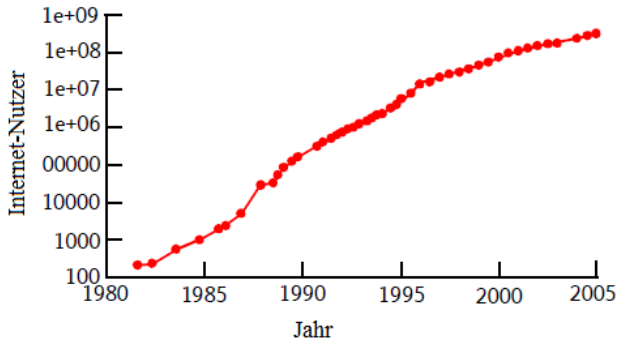


Abbildung 1: Wachstum der Internet-Nutzung, Zitat nach [18]

3.1.1 Fehlerkonfigurationen

Das Internet beinhaltet eine große Anzahl von autonomen Systemen, die unter sich Routing-Informationen austauschen und somit die besten Wege zu verschiedenen Destinationen lernen können. Derzeit wird das *Border Gateway Protocol (BGP)* [4] als Standard-Protokoll bei dem Interdomain-Routing eingesetzt und ist somit ein wichtiger Bestandteil des Alltags geworden. BGP gehört zu der Klasse der *Pfadvektorprotokolle* und dabei tauscht jeder BGP Router mit seinen Nachbarn die gesamte Liste aller *AS-Path* Informationen. Um überhaupt Routing-Informationen austauschen zu können, werden sogenannte *Peering-Sessions* aufgebaut. Dabei werden die kompletten Routing-Tabellen ausgetauscht um somit ein globales Routing zu ermöglichen. Nach dieser Anfangsphase werden meistens Update-Nachrichten verschickt, die eine mögliche Route-Änderung signalisieren sollen. Allerdings besteht die Gefahr, dass ein Router bzw. ein Routerinterface schnell zwischen „up“ und „down“ Zustand wechselt und dabei R*oute-Flapping* [5] erzeugt. Der Grund dafür wäre beispielsweise eine Fehlerkonfiguration, die zur Überschwemmung eines Routers mit Update-Nachrichten führen kann. Dabei wäre eine Überlastung durch ständiges Löschen und Einfügen von Einträgen in der Routing-Tabelle möglich.

Ein ähnliches Phänomen ist auch am 16 Februar 2009 aufgetreten, als die Inkompetenz eines Administrators des tschechischen Providers „SuproNet“ einen globalen *Buffer-Overflow* erzeugt hat [6]. Grund war eine künstliche Verlängerung des AS-Pfades, die bei vielen von Cisco IOS betriebene Router einen Verbindungsabbruch und Neustart verursacht hat.

3.1.2 Reaktionszeiten

Das Ziel der Routing-Protokolle ist nicht nur den optimalen Weg zwischen zwei Netzknoten zu bestimmen, sondern auch möglichst schnell auf Topologieänderungen zu reagieren. Mathematisch ausgedrückt soll ein Protokoll gute Konvergenzeigenschaften aufweisen und zusätzlich Skalierbar sein.

Auch wenn durch die Aufteilung des Internets in autonome Systeme eine bessere Skalierbarkeit erreicht wird, so bleiben noch einige Konvergenzprobleme des BGP bestehen, die zu untersuchen sind. Im Laufe der Zeit wurden unterschiedliche Techniken entwickelt, die die Anzahl von Update-Nachrichten verringern sollen. Beispielsweise kommt *Route-Flap-Damping* zum Einsatz um die Probleme von *Route-Flapping* zu lösen. Dabei verfügt jeder BGP-Router über einen Penalty-Wert, der die Instabilität der Route messen soll. Dieser Wert wird inkrementiert wenn z.B. eine Route ständig zwischen verfügbar und nicht verfügbar pendelt. Es wird dann auf lokaler Ebene entschieden, wann die Route wiederverwendet werden soll (z.B. mit Hilfe eines Schwellwertes). Allerdings kann *Route-Flap-Damping* ein schlechtes Konvergenzverhalten aufweisen und somit Verzögerungen bis zu einer Stunde verursachen [5].

MRAI-Timer ist ein weiterer Mechanismus, mit dem BGP versucht, die Stabilität des Routings zu erhöhen. Jedes Mal wenn ein Router ein Update bezüglich eines bestimmten Präfixes an einen Nachbarn schickt, wird der Timer gestartet. Erst wenn er abgelaufen ist, können wieder Update-Nachrichten geschickt werden. Dadurch werden *Update-Bursts* verhindert. Meistens hat der Timer einen Standardwert von 30 Sekunden. Allerdings haben Tarek Sobh, Khaled Elleithy and Ausif Mahmood [7] gezeigt, dass ein Timer-Wert von 2-5 Sekunden zu Ausfällen und somit zu schlechter Konvergenz führen kann.

3.2 Dienstabhängigkeit im Internet

Einen großen Einfluss auf die Resilienz hat auch die Dienstabhängigkeit im Internet. Beispielsweise werden Google-Dienste von mehreren hundert Millionen Menschen weltweit genutzt. Laut dem Statistischen Bundesamt bestellten 29,5 Millionen Menschen in Deutschland im ersten Quartal 2009 Waren und Dienstleistungen über das Internet. Dabei ist Google eine der Hauptquellen für weitergeleiteten Verkehr im Internet, und ein längerer Ausfall der Seite würde viele Geschäfte beeinflussen.

Ein weiterer Faktor wäre die Positionierung von DNS-Servern im Netz eines Unternehmens. Ein klassisches Beispiel ist die DoS-Attacke auf die Router, die für den Datenverkehr zu den Microsoft-Webseiten zuständig waren. Da im Jahre 2001 die DNS-Server von Microsoft im gleichen IP-Subnetz standen (die IP-Adressen 207.46.138.11, 207.46.138.12, 207.46.138.20 und 207.46.138.21), wurde dadurch der Zugang zu einigen Internet-Angeboten, darunter microsoft.com, microsoft.de, msn.com und WindowsMedia.com teilweise verhindert.

3.3 Sicherheit

Im Internet kommt der Sicherheit des eigenen Netzwerks oder des eigenen Rechners eine besondere Bedeutung zu. Während früher nur Netzwerke der Universitäten und Forschungs-einrichtungen anfällig an Angriffen waren, ist heute praktisch jeder der Gefahr ausgesetzt, durch fehlende Sicherheitskonzepte Opfer zu werden - mit oft fatalen Folgen.

In LANs, in denen das *Address Resolution Protocol (ARP)* eingesetzt wird, sind sehr viele wirkungsvolle Spoofing-Angriffe möglich. Es reicht dann meist schon, die Kontrolle über einen der Rechner im LAN zu bekommen, um das gesamte Netz zu kompromittieren. Dabei kann es zum sogenannten *Man-in-the-middle-Angriff* kommen, bei dem der Angreifer vollständige Kontrolle über den Datenverkehr zwischen zwei oder mehreren Netzknoten erlangen kann. Mögliche Angriffsstrategien wären das Modifizieren der ARP-Tabellen der Opfersysteme oder das Vorspielen eines falschen DHCP-Servers. Neulich hat das Unternehmen „RedTeam Pentesting“ es geschafft beim Online-Banking Überweisungen unbemerkt auf dritte Konten umzuleiten [8].

Ein weiteres Problem wären die fehlenden Autorisierungs-mechanismen bei BGP. Ein BGP-Router würde beim Eintreffen eines Datenpackes seine Routingtabelle nach der besten Route zum Ziel durchsuchen. Der Router wird den kürzesten Weg auswählen und danach die Daten schicken. Allerdings vertrauen Router darauf, dass ihnen alle anderen Router die Wahrheit sagen. Wenn ein Router mitteilt, dass er die kürzeste Route zu einem bestimmten Ziel kennt, dann entspricht es der „Wahrheit“, und die Pakete werden diesem Router übergeben. Möchte ein Angreifer den Verkehr zu einem bestimmten System umleiten, muss er nur die Routing-Einträge manipulieren, indem er behauptet eine kürzere Route zum Ziel zu kennen. Diese Strategie ist noch in der Welt der Netzwerke als *IP-Hijacking* bekannt. Im Februar 2008 wurde von der „Pakistan Telecom“ eine Anweisung zum Sperren des Zugriffs auf YouTube umgesetzt, indem auf dem Border-Gateway eine spezifischere Route zu den Servern von YouTube angekündigt wurde. Die empfangenen Pakete wurden einfach verworfen. Allerdings übernahm der Upstream-Provider von „Pakistan Telecom“ diese Information und verbreitete sie im Internet. Danach wurden alle an YouTube adressierten Daten an das Null-Device eines Rechners von „Pakistan Telecom“ ausgeliefert [9].

4. Resilienz-Techniken

Obwohl viele Internet-Nutzer das heutige Internet als anständig funktionierend einschätzen, wurde im Abschnitt 3 klar gezeigt, dass es leider nicht als Robust und Widerstandsfähig eingestuft werden kann. Um ein angemessenes QoS-Niveau garantieren zu können, wurden zahlreiche Resilienz-Techniken erfunden, die Thema dieses Abschnittes sind.

Resilienz-Techniken können in zwei Klassen aufgeteilt werden. Einerseits werden proaktive Mechanismen eingesetzt, die eine zukünftige Störung oder Ausfall vermeiden sollen. Dabei wäre folgende hypothetische Situation denkbar: jeder Netzknoten erstellt eine Routing-Tabelle, in der ein Weg zu jedem möglichen Zielknoten steht. Es werden periodisch Tabelleninformationen zu den Nachbarn des jeweiligen Knotens geschickt. Auf dieser Weise ist eine Route zu jedem Netzteilnehmer bekannt. Andererseits sind auch reaktive (on-demand) Techniken ein möglicher Ausweg aus Fehlersituationen: ein Weg zu einem Knoten wird nur gesucht, wenn auch ein Datenpaket verschickt werden soll.

4.1 Protection Switching

Protection Switching wird in der Welt der Netzwerke als die Möglichkeit bezeichnet, in Fehlersituationen auf redundante (engl. backup) Pfade umschalten zu können. Dabei zählt sie zu den proaktiven Mechanismen auf Schicht 2 des ISO-OSI Modells und kann sowohl bei Leitungsvermittelnde (engl. circuit switching) Netzen, als auch bei Paketvermittelnde (engl. packet switching) Netzen eingesetzt werden.

Bei dieser Technologie unterscheidet man hauptsächlich zwischen *Hauptpfaden*, durch denen die eigentlichen Daten geschickt werden, und *Ersatzpfaden*, die im Fall eines Linkausfalls oder Störung als Backup verwendet werden. Dabei würde man meistens eine Ring-Topologie aufbauen, da im Falle eines Fehlers schnell in der entgegengesetzten Richtung umgeschaltet werden kann.

Ein mögliches Arbeitskonzept zur Realisierung des Mechanismus wäre das Einfügen von sogenannten Domänen, die eine Ring-Struktur aufweisen. Dabei besitzt jede Domäne einen Master-Knoten und zahlreiche Transit-Knoten. Jeder Knoten ist mit zwei Ports an dem Ring angeschlossen, wobei der eine als *Primary-Port* und der andere als *Secondary-Port* gekennzeichnet ist. Im normalen Betrieb blockiert der Master-Knoten seinen Secondary-Port für alle Datenpakete mit Ausnahme der Prüfpakete um somit eine Schleifenbildung zu verhindern. Wenn eine Transit-Station einen Linkausfall entdeckt, so wird es den Master-Knoten mitgeteilt. Dabei wird der Secondary-Port des Masters geöffnet um somit der Richtungswechsel zu ermöglichen. Durch den Richtungswechsel wird ein anderer Pfad genommen, der noch als Backup bezeichnet wird.

Allerdings ist es möglich, in einem Netz von Punkt A bis Punkt B mehrere Ersatzpfade zu haben oder sogar Haupt- und Ersatzpfad gleichzeitig zu verwenden. Hierbei unterscheidet man zwischen fünf Schutzmechanismen:

1+1 Protection: Bei diesem Schutzmechanismus wird ein Hauptpfad durch einen knoten- und kantendisjunkten Ersatzpfad geschützt. Dabei werden die Daten gleichzeitig über beide Pfade geschickt um im Falle einer Störung des Hauptpfades können diese am schnellsten zum Ziel gelangen. Allerdings kann man den Ersatzpfad nur für den Backup-Traffic verwenden und man hat beispielsweise nicht die Möglichkeit Traffic niedrigerer Priorität dadurch zu transportieren. Zusätzlich ist ein Overhead von mindestens 100 % garantiert.

1:1 Protection: Bei diesem Konzept wird wieder ein Hauptpfad durch einen knoten- und kantendisjunkten Ersatzpfad geschützt. Allerdings wird der Ersatzpfad für den Backup-Traffic nur dann verwendet, wenn eine Störung vorliegt. Wie man sich leicht vor Augen führen kann, kann der Backup-Pfad für den Transport niedrig priorisierter Daten verwendet werden. Wenn aber ein Link- oder Knotenausfall vorkommt, wird das Senden des Datenstroms angehalten um somit für den hoch priorisierten Verkehr „Platz zu schaffen“.

M:N Protection: Der M:N Mechanismus wird als eine Weiterentwicklung des 1:1 betrachtet. Hierbei werden N Hauptpfade durch M Ersatzpfade geschützt mit der Nebenbedingung, dass meistens $M \leq N$ gilt. Im Falle, dass mehr als M Hauptpfade gestört werden, kann eine sichere und korrekte Datenübertragung nicht garantiert werden. Es besteht auch die Möglichkeit für N Hauptpfade nur einen Backup-Pfad zu reservieren. Dies entspricht dem 1:N Mechanismus, der noch als Spezialfall des M:N Mechanismus zu betrachten ist. Allerdings ist es nicht möglich den einen Ersatzpfad von mehreren Hauptpfaden gleichzeitig benutzen zu lassen.

Demand-Wise Shared Protection (DSP): Dieses Konzept eignet sich gut für optische Netzwerke, da man leicht die Bandbreite einer Verbindung aufteilen und für verschiedene Zwecke reservieren kann [10]. Sein Vorteil basiert auf einer effizienteren Ausnutzung der Netzstruktur, so dass die Anzahl notwendiger Ersatzwege reduziert werden kann. Allerdings hat man keine Separation zwischen Haupt- und Ersatzpfaden, sondern vielmehr eine Unterteilung der Bandbreite. Es wäre also möglich bei Bedarf einen Pfad gleichzeitig für den Haupt- und Backuptraffic zu verwenden. Im Fall eines Knoten- oder Linkausfalls wird der entsprechende Datenverkehr über den anderen Pfaden verteilt.

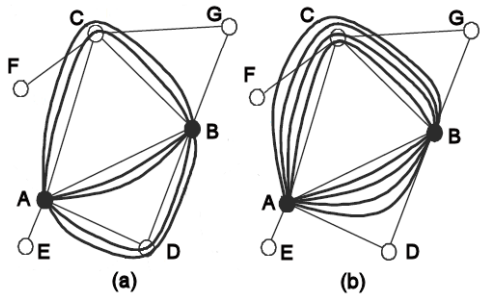


Abbildung 2: Demand-Wise Protection und 1+1 Protection

Ein Vergleich zwischen *1+1* und Demand-Wise Shared Protection ist in der Abbildung 2 zu sehen. Es müssen insgesamt vier Dateneinheiten von A bis B gleichzeitig transportiert werden. Allerdings muss sichergestellt werden, dass jeder datentransportierende Pfad gegen einzelne Link- und Knotenausfälle geschützt ist. Im Falle der DSP-Strategie (a) werden insgesamt 6 Pfade benötigt um die Anforderungen zu erfüllen. Bei der *1+1* Technik (b) werden 8 Pfade benötigt, was zu höheren Vernetzungskosten führen kann. Wie in Referenz [11] diskutiert wird, sind die beiden Techniken zusätzlich im Punkt Dienstverfügbarkeit vergleichbar.

Protection Cycles: Diese Sicherheitstechnik wird noch *P-Cycles* genannt und wird meistens in ringförmigen optischen Netzwerken verwendet, da diese oft an Linkausfälle anfällig sind. Wie in der Abbildung 3(a) zu sehen ist, wird im Voraus ein P-Cycle konfiguriert um somit eine gewisse Anzahl von Kanten und Knoten zu schützen.

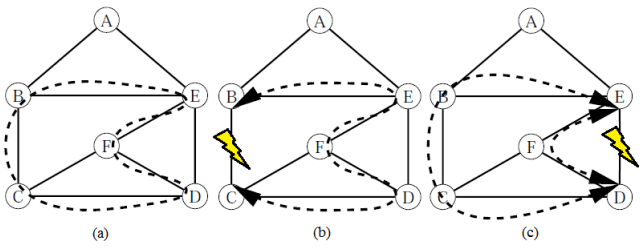


Abbildung 3: Protection Cycles

Die Links, die im Zyklus B-C-D-F-E-B zu finden sind, werden noch *On-Cycle-Links* genannt. Im Falle eines Ausfalls (Abbildung 3(b)) wird es immer noch möglich sein den Verkehr auf den Rest des Zyklus C-D-F-E-B weiterzuleiten. Zusätzlich werden Links, deren Endpunkte auf dem Zyklus liegen geschützt. Die werden noch *Straddling-Links* genannt. Ein Beispiel dazu ist in der Abbildung 3(c) zu sehen, bei dem der Link E-D ausgefallen ist. Damit der Datenaustausch zwischen den Knoten E und D funktionieren soll, werden die alternativen Routen E-F-D oder E-B-C-D verwendet.

4.2 Multiprotocol Label Switching

Im Abschnitt 3.1.2 wurde bereits die Problematik geschildert, dass das Routing heute zwar auf Knoten- und Linkausfälle reagieren kann, aber dabei schlechte Konvergenzeigenschaften aufweisen kann. Zusätzlich muss man betonen, dass die heutige Netzarchitektur komplexe Entscheidungen, Auswertungen und Funktionen den Endgeräten überlassen hat. Dabei wurde die Leistungsfähigkeit eines Zwischenknotens (Router, Switch) beschränkt, was auch zu Problemen führen kann. Da das heutige Internet als verbindungsloses Netzwerk arbeitet, muss jeder Router selber Entscheidungen treffen wie ein gewisses Paket weitergeleitet werden soll. Meistens werden Pakete in *Forwarding Equivalence Classes (FEC)* eingeteilt (z.B. mit gemeinsamer Zieladresse) um dann im nächsten Schritt alle mögliche Folgerouter auf diese FEC abzubilden. Auf dem Weg durch ein Netzwerk wird das Paket an jedem Router erneut untersucht und einer FEC zugeordnet.

Multiprotocol Label Switching (MPLS) bietet seit Ende der 1990er die Möglichkeit, Router zu entlasten um somit die verfügbaren Bandbreiten des Netzes besser auszulasten. Die Grundidee ist es, nicht mehr Pakete Hop-by-Hop durch ständigen Tabellen-Lookup weiterzuleiten, sondern diese an einem Eingangspunkt (Ingress-Router) auf einem vorbestimmten Datenpfad zu schicken und erst wieder an einem Ausgangspunkt (Egress-Router) das Standard-Forwarding zu verwenden. Meistens wird MPLS in Netzen verwendet, die auf einer IP-basierten Struktur aufweisen. Dabei besteht die Möglichkeit ein ganzes AS als eine MPLS-Domäne zu betrachten um somit Traffic schnell innerhalb Providergrenzen zu routen.

Typescherweise verlässt man sich bei der Topologiebestimmung auf ein *Interior Gateway Protocol (IGP)*, wie IS-IS oder OSPF. Danach werden *Label Switched Paths (LSPs)* aufgebaut, deren Anfang ein Ingress-Router und Ende ein Egress-Router ist. Dabei werden meistens ein Anfangs- und ein Endknoten eines AS gewählt. Jetzt kann beispielsweise eine FEC Zuordnung nur ein Mal vom Ingress-Router durchgeführt werden. Sobald ein Paket ein MPLS-Netz betritt, wird es mit einem 32 Bit MPLS-Header [17] versehen, der u.a. den Weg durch das Netz bestimmen soll (Abbildung 4).

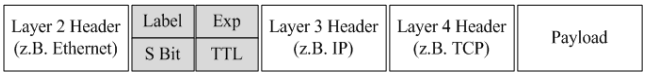


Abbildung 4: Header-Reihenfolge bei MPLS

Alle entlang eines LSP liegenden Router (Label Switch Router, LSR) treffen anhand des im Header befindbaren Labels ihre Forwarding-Entscheidung. Hier liegt auch der große Unterschied zu IP, da durch die Beförderung entlang eines LSP keine lokal bestimmte Wegewahl mehr stattfindet. Beim Forwarding wird an einem Router nun der Wert des jeweiligen Labels betrachtet um zu einer Entscheidung zu gelangen. Es ist nicht mehr nötig wie bei dem Longest Prefix Match-Forwarding den ganzen IP-Header

zu empfangen, der in der Regel mehr Informationen als gebraucht enthält. Dabei ist das Label nur eine Informationseinheit, die beispielsweise ein Paket zur jeweiligen FEC zuordnet.

MPLS bietet zusätzlich den *Fast-Reroute-Mechanismus (MPLS-FRR),* der bei Bedarf Traffic umleiten kann um somit die Resilienz des Netzwerkes zu erhöhen. Um die negativen Konsequenzen eines Link- oder Knotenausfalls zu vermeiden, werden neben den LSPs an jedem Knoten zusätzliche Backup-Pfade aufgestellt, über welche die Datenpakete umgeleitet werden sollen. Dabei wird der Traffic von dem jeweiligen Knoten (Point of Local Repair, PLR) auf den unmittelbaren Nachbarn weitergeleitet. Der Merge-Point ist frei wählbar und kann beispielsweise der Egress-Router sein (Abbildung 5).

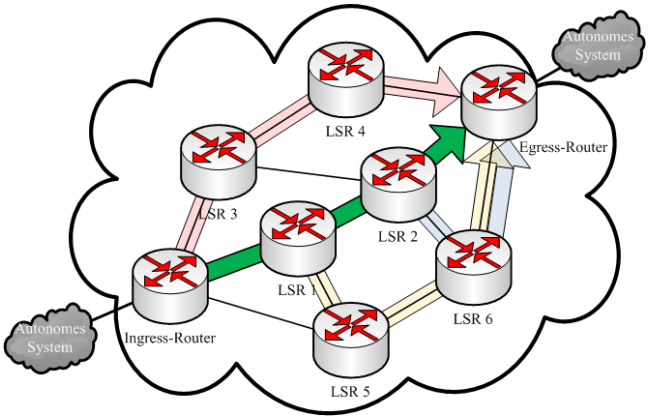


Abbildung 5: MPLS-FRR mit grünem LSP und rotem, gelbem, blauem Backup

Es existieren zwei verschiede Arten, den FRR-Mechanismus zu realisieren. In der Praxis unterscheidet man hauptsächlich zwischen den *One-to-One* und den *Many-to-One* Ansatz. Wie bei Protection Switching, kann hier die Anzahl der Backup-Pfade vom Provider bestimmt werden um ein akzeptables Niveau zwischen Backup-Aufwand und angebotene Leistung zu finden. Bei dem One-to-One Konzept verwalten die PLR-Knoten separate Backup-Pfade für jeden LSP, der durch denen passiert. Um allerdings einen drohenden Verkehrsengpass [12] zu verhindern, wird meistens die Many-to-One Methode eingesetzt, bei der ein Ersatzpfad mehrere LSPs schützen darf. Dabei wird die Anzahl der Update-Nachrichten, die zur Verwaltung der Backups nötig sind, enorm reduziert.

Ein komplett anderer Lösungsweg wäre auch die Benachrichtigung des Ingress-Routers im Falle einer Störung. Dann wäre es durchaus möglich eine neue LSP zu wählen um die Datenpakete zum Ziel zu bringen. Allerdings ist es für Provider eine ungünstige Situation, da sich im Rahmen eines Netzes mehrere tausend Knoten befinden können. Der FRR-Mechanismus würde in einer Fehlersituation maximal 50 ms [13] brauchen, da man nur auf den unmittelbaren Nachbarn den Datenverkehr umleiten muss.

4.3 Content Distribution Network

Auch wenn heutzutage die Internet-Nutzer immer schnellere Zugriffsmöglichkeiten bekommen und die Backbone-Netzen sich rapide verbessert haben, kann es beim Zugriff auf Webinhalte zu hohen Verzögerungen kommen. Die Server sind teilweise überlastet oder überhaupt nicht mehr zu erreichen und die Dienstgüte ist nur begrenzt anzubieten. Manchmal werden diese negativen Auswirkungen auch ironisch als *World Wide Wait* bezeichnet.

Um dem entgegenzuwirken wird eine zuverlässige, skalierbare und hohes QoS-Niveau garantierende Technik benötigt. Ein möglicher Losungsweg wäre der Einsatz von *Content Distribution Networks (CDNs)*, die eine resiliente und schnelle Auslieferung von Webinhalten bieten können. Allgemein enthält ein solches Netz mehrere Server, die weltweit an verschiedenen Orten verteilt sind. Deren Anzahl und Positionierung hängt allerdings alleine vom CDN-Provider ab. Der weltweit größte Anbieter für Auslieferung und Beschleunigung von Online-Anwendungen „Akamai“ beinhaltet beispielsweise 40.000 Server, die in 70 Länder zu finden sind [14]. Dabei beinhalten alle CDN-Knoten dieselben Datenkopien und sind strategisch im Netz platziert um niedrigere Latenzzeiten, hohe Robustheit und Zuverlässigkeit anzubieten. Das CDN hat dann die Möglichkeit, einem Endnutzer zu einem für ihn am besten geeigneten Server umzuleiten, anstatt auf einen Hauptserver zuzugreifen. Den großen Erfolg haben CDNs allerdings den *Content Delivery* und *Content Routing* Komponenten zu verdanken, die im Folgenden untersucht werden.

Content Delivery (Abbildung 6) beschäftigt sich mit der Kodierung, Speicherung, Bereitstellung und Auslieferung der Inhalte. Um Überlastsituationen zu vermeiden, wurde eine Struktur mit Cache-Server, die an strategischen Punkten im Netz (Points-of-Presence, POPs) zu finden sind, entwickelt. Es existieren zwei verschiedene Methoden, die die Versorgung eines Servers mit Dateninhalten ermöglichen. Man unterscheidet zwischen der *Pre-* und *Just-in-Time-Caching*. Besonders elegant ist die Pre-Caching Methode, da die Inhalte im Voraus vom Hauptserver ausgeliefert werden. Meistens wird diese Verteilung zu Niedriglastzeiten durchgeführt, um somit keinen Einfluss auf die Auslieferungszeiten zu haben. Man kann sich allerdings auch auf das Just-in-Time-Caching verlassen, dass den Dateninhalt nur dann besorgt, wenn auch eine Anfrage von einem Kunden vorliegt.

Der Einsatz des Internet Cache Protokolls (ICP) [15] wäre im Zusammenhang mit den genannten Techniken auch denkbar. Wenn beispielsweise ein Server eine Anfrage für ein Objekt erhält, dass nicht verfügbar ist, kann er mit Hilfe des Protokolls die gewünschten Daten von anderen Cache-Server empfangen, ohne den Hauptserver zu befragen.

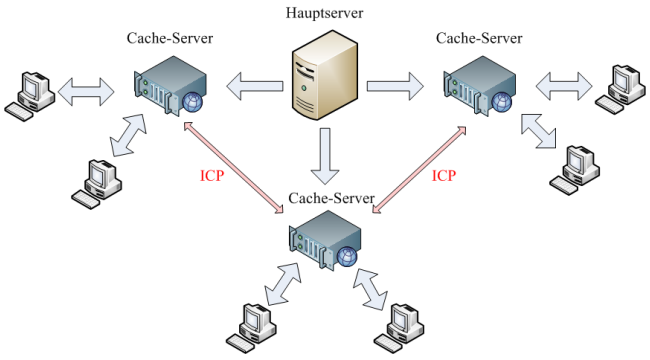


Abbildung 6: Cache-Struktur bei CDN

Content Routing (Abbildung 7) befasst sich allerdings damit, den möglichst besten Cache-Server zu finden, um Dateninhalte an den Kunden auszuliefern. Wenn ein Kunde einen Dateninhalt von der Seite eines Content-Anbieters herunterladen will, wird er normalerweise auf den DNS-Server eines CDNs weitergeleitet.

Der Client-Computer kann nun an einen für ihn am besten geeigneten Cache-Server weitergeleitet werden.

Dieses Forwarding kann nach unterschiedlichen Gesichtspunkten geschehen. Beispielsweise kann die geographische Nähe oder Antwortzeit als Hauptkriterium genommen werden. Ein anderer möglicher Lösungsweg wäre auch die Wahl des Servers, der im Subnetz des Client-Computers liegt.

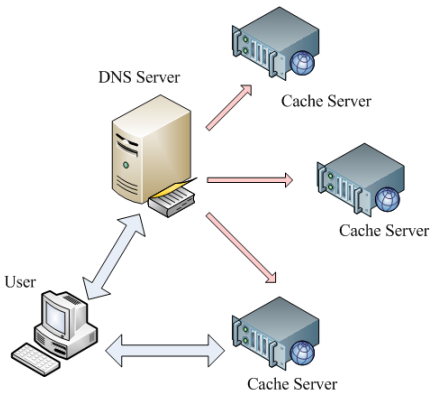


Abbildung 7: CDN-Routing

Eine weit verbreitete Methode, die ein *Load-Balancing* auf Server mit gleichem oder ähnlichem Inhalt erreicht, ist der Einsatz von sogenannten Layer 4-7 Switches. Bei einem CDN-Netzwerk heißt dies, dass hinter einem Point-of-Presence mehrere Knoten stehen, die mit identischem oder gleichem Dateninhalt ausgestattet sind. Der Switch fungiert als Proxy für alle Anfragen an diese Server und stellt eine virtuelle IP-Adresse für den gesamten POP zur Verfügung. Wenn der Switch plötzlich merkt, dass in einer Anfrage mehrere Objekte angefordert werden, darf er der Reihe nach die Server befragen. Falls ein Objekt auf mehreren Servern zu finden ist, wird der Switch den am wenigsten ausgelasteten Server auswählen und von diesem die gewünschte Dateien anfordern.

4.4 Peer-to-Peer-Netze

Peer-to-Peer-Netze (P2P) haben sich in den letzten Jahren eine rasant wachsende Popularität erfahren. Der größte Vorteil dieses Konzepts gegenüber den bekannten Client-to-Server Modell liegt in der möglichen Skalierbarkeit zur Unterstützung von Tausenden von Nutzern sowie Erzielung von Verlässlichkeit durch Dezentralität. Meistens wird P2P als ein sich selbst organisierendes System gleichberechtigter Einheiten (Peers) bezeichnet, dass ohne Nutzung zentraler Ressourcen operiert. Dabei werden Bandbreite, Speicherplatz, Rechenkapazität möglichst gleichmäßig verteilt genutzt.

Im Unterschied zu den zentralen Server-Architekturen, bei denen der Speicherort einer Datei bekannt ist, können Daten in dezentralen Systemen an zahlreichen Stellen im Netz gespeichert werden. In den letzten Jahren haben sich zur Lösung dieses Problems zwei Richtungen entwickelt, die Thema dieses Abschnittes sein werden.

Zum einen wurden die sogenannten *unstrukturierten P2P-Systeme* entwickelt, die heutzutage als die erste P2P-Generation bezeichnet werden. Meistens verlassen sich solche Systeme auf einem Hauptknoten, der die Lokation jeder Datei im Netz kennt. Damit können Peers erst nach einer Lokationsanfrage miteinander kommunizieren. Es existieren auch andere Ansätze, die das Prinzip des Flutens nutzen. Da im unstrukturierten P2P prinzipiell jeder Knoten jeden anderen Knoten als Nachbarn wählen kann, wird eine Suchanfrage weiterpropagiert um möglichst viele Peer-Knoten, die im Besitz der Datei sind zu finden. Allerdings kann es bei einer großen Anzahl von Zwischenknoten zu Überlastsituationen kommen. Zusätzlich ist keine Garantie vorhanden, dass die Suchaktion erfolgreich sein wird. Normalerweise werden Pakete mit einem Hop-Zähler ausgestattet um ein endloses Herumirren und unnötiger Ressourcenverbrauch zu vermeiden. Anschließend lässt sich sagen, dass solche Systeme auch anfällig an DoS-Attacken sind. Wie in Referenz [20] diskutiert, können bösartige Knoten andere erzwingen, einen fremden Host anzusprechen um somit Zugriff auf unerwünschte Dateien zu ermöglichen.

Zum anderen wurden *strukturierte P2P-Systeme* entwickelt, die die Probleme der Skalierbarkeit und Effizienz lösen sollen. Dabei besitzen diese eine logische Struktur und verwenden meistens den Ansatz der *Distributed Hash Tables (DHT)*, der das Auffinden einer Datei mit einer Komplexität von O(log N) ermöglicht. Da zentralisierte Tabellen oftmals langsam und anfällig gegen Angriffe sind, wird eine Aufspaltung der Hash-Tabelle durchgeführt und von verschiedenen Knoten verwaltet. Dabei wird jeder Datensatz durch einen eindeutigen Schlüssel identifiziert, der mittels einer Hash-Funktion berechnet wird. Wenn beispielsweise eine Datei gespeichert werden soll, wird mit Hilfe dieser Funktion ein solcher Wert generiert und zum verantwortlichen DHT-Knoten propagiert. Damit kann dieser Knoten in seiner Tabelle einen Zusammenhang zwischen der Datei und ihren Hash-Wert herstellen. Dieser Ansatz würde somit ein schnelles Auffinden ermöglichen, da wenn ein Peer eine bestimmte Datei haben will, kann er leicht die globale Hash-Funktion verwenden und mit Hilfe des resultierenden Wertes die verschiedenen DHT-Knoten befragen. Dabei wäre jeder DHT-Knoten in der Lage schnell eine Antwort zu generieren ob die gewünschte Datei in seinem „Bezirk“ ist.

Manche Verfahren [21] bieten auch die Möglichkeit, die verschiedenen Hash-Werte an der Netz-Topologie anzuordnen. Um das zu realisieren, wird jedem Peer eine ID zugeordnet, die beispielsweise ein Hash-Wert der IP-Adresse sein kann. Dabei existiert eine injektive Abbildung zwischen den Schlüssel-Raum und die Menge aller IDs, die die Grundlage bei einer zukünftigen Suche sein wird. Meistens werden einem Peer die Schlüssel zugeordnet, die nah an seiner ID sind. Dadurch werden Antwortzeiten auf einer Anfrage senken, da nur ein Vergleich mit der eigenen ID nötig wäre.

5. Ausblick und weitere Tendenzen

Das Internet ist mittlerweile zu einer großen Baustelle geworden. Viele behaupten, dass es sich sogar in der Krise befindet. Adressknappheit, Sicherheitsprobleme, Skalierungsprobleme beim Routing, Spam, Cyber-Kriminalität sind zur Motivation von Entwicklungsgruppen geworden komplett neue Ansätze zu entwickeln und einzusetzen. Somit wurde das Einführen von neuen Protokollen und Techniken, die die Netzwerk-Resilienz deutlich verbessern sollen, zum Thema vieler Forschungsarbeiten [16].

Ziel dieses Abschnittes ist den Leser einen kleinen Ausblick über zukünftige Methoden und Tendenzen, die sich heutzutage noch im Entwicklungsstadium befinden, zu geben.

Mittlerweile ist die Einführung des *IPv6-Protokolls* zum Lieblingsthema vieler Forscher und Entwickler geworden. Man hat schnell gemerkt, dass in Regionen, in denen sehr viele Menschen wohnen und wo gleichzeitig ein gewaltiger Wirtschaftsaufschwung stattfindet, es zu einer Knappheit im

Bereich der IPv4 Adressen kommen kann. Das IPv6 Protokoll wird diese Probleme lösen können, da es eine Erweiterung des Adressraums von 32 auf 128 Bit anbietet. Mit dieser Technologie werden ca. 340 Sextillionen zur Verfügung stehen, was auch wahllose Scans über den gesamten Adressbereich vermeiden wird. Der Nachfolger von IPv4 wird auch neue Eigenschaften wie Mobile IPv6, QoS-Unterstützung, IPsec-Verschlüsselung, Multicast und allgemeine Verbesserungen des Protokollrahmens mit sich bringen können [13].

DNSSEC ist eine weitere Tendenz, die auch als Erweiterung des Domain Namen Systems (DNS) bezeichnet wird. Sie dient dazu, die Authentizität und die Vollständigkeit der Daten von DNS-Antworten sicherzustellen. DNSSEC ist eine Art Versicherung, die einem Benutzer garantieren kann, dass nur diejenige Webseite angezeigt wird, die er aufrufen will. Dabei werden die DNS-Anfragen und -Antworten mit Hilfe kryptografischer Unterschriften gegen Verfälschungen gesichert [13]. Es ist auch möglich die Echtheit des kontaktierten Servers durch das SSL (Secure Socket Layer) Protokoll zu garantieren. Allerdings soll DNSSEC verhindern, dass man nicht schon auf einem falschen Server landet, bevor die Verbindung durch SSL gesichert wird.

Das *Stream Control Transmission Protokoll (SCTP)* kann als möglicher Nachfolger für das TCP und teilweise UDP angesehen werden [19]. Eine essenzielle Eigenschaft des Protokolls ist die Unterstützung von Multihomed-Knoten. Diese können unter verschiedene IP-Adressen erreicht werden und man hat die Möglichkeit Pakete über mehrere Wege zu schicken. Dabei erhöht man die Netzwerk-Resilienz, da auch im Falle einer Leitungsstörung die SCTP Pakete beim Empfänger ankommen können.

6. Zusammenfassung

Das Internet ist mittlerweile zur Grundlage der modernen Industriegesellschaft und der globalen Wirtschaft geworden. Gleichzeitig besitzt es eine große Bedeutung für den privaten Bereich. Es basiert jedoch größtenteils auf Techniken und Algorithmen, die in den 70er und 80er Jahren entwickelt wurden, die allerdings die heutigen Anforderungen an Sicherheit, Flexibilität und Zuverlässigkeit teilweise oder komplett nicht erfüllen können. Viele behaupten, dass es sogar zum Opfer seines Erfolgs geworden ist: Telefonate und Videos, Live-Übertragungen und Software-Downloads belasten das Kommunikationsnetz. Schlecht konzipierte Übertragungsnetze können schwerwiegende Folgen für die Dienstgüte haben. Manchmal kann die Wiederherstellung des Dienstes statt nur Sekunden mehrere Stunden in Anspruch nehmen. Längere Netzausfälle führen zu Dienststörungen, die die heutigen Geschäfte beeinflussen können. Es existieren leider zahlreiche technische Gründe, die im Abschnitt 3 erwähnt wurden, die gegen ein Widerstandsfähiges Internet sprechen.

Dennoch funktionieren die heutigen Netzwerke erstaunlich gut. Dies ist allerdings den verschiedenen Resilienz-Techniken zu verdanken, die ein angemessenes QoS-Niveau garantieren können. Manche Strategien, wie CDN und P2P-Netze, versuchen von der ossifizierten und standardmäßigen Netz-Architektur abzuweichen um somit ein gegen unerwartete Störungen widerstandsfähiges Netzwerk zu konzipieren. Dabei wurde auch das Thema Performanz und Effizienz angesprochen, die mittels geeigneter Last-Balance und Verteilung der Daten erheblich verbessert werden können. Andere Techniken, wie Protection Switching, verlassen sich auf die Redundanz, die im Falle eines Pfadausfalls ein schnelles Umleiten des Datenverkehrs ermöglicht. Man darf zusätzlich Tunneling-Mechanismen nicht unterschätzen, die zur Erhöhung der Resilienz und der Performanz führen können. Techniken, wie MPLS, bieten nicht nur ein schnelles Umleiten des Datenverkehrs, sondern auch eine Entlastung der Zwischenknoten.

Anschließend lässt sich sagen, dass Network Resilience auch im zukünftigen Internet ein aktives Forschungsthema sein wird. Techniken, wie IPv6, DNSSEC und SCTP, sind nur ein kleiner Teil der heutigen Forschungsgebiete, die allerdings eine spannende Zukunft versprechen.

7. Referenzen

[1] FP7 – 224619*, Resilience and Survivability for Future Networking*, http://www.resumenet.eu/, eingesehen am 15.03.2010.

[2] MS03-026, *Microsoft Security Bulletin*, http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx, eingesehen am 15.03.2010.

[3] J. P. G. Sterbenz and D. Hutchison, *ResiliNets Wiki*, https://wiki.ittc.ku.edu/resilinets_wiki/index.php/, eingesehen am 16.03.2010.

[4] T. Schwabe, *IP-Netze mit Interdomain-BGP-Routing: Konvergenzverhalten, Dienstqualität und Dimensionierung*, Technische Universität München, September 2006, Seite 7-16.

[5] Z. Mao, R. Govindan, G. Varghese and R. Katz, *Route Flap Damping Exacerbates Internet Routing Convergence*, SIGCOMM, Pittsburgh – USA, August 2002.

[6] Renesys Blog, *Reckless Driving on the Interne*, http://www.renesys.com/blog/2009/02/the-flap-heard-around-the-worl.shtml, eingesehen am 17.03.2010.

[7] T. Sobh, K. Elleithy and A. Mahmood, *Novel Algorithms and Techniques in Telecommunications and Networking,* Chapter 17: Improving BGP Convergence Time via MRAI Timer, Springer 2010.

[8] RedTeam Pentesting GmbH, *Man-in-the-Middle-Angriffe auf das chipTAN comfort-Verfahren im Online-Banking*, Aachen, November 2009.

[9] RIPE NCC, *YouTube hijacking: A RIPE NCC RIS case study*, http://www.ripe.net/news/study-youtube-hijacking.html, eingesehen am 15.03.2010.

[10] A. Zymolka, A. Koster and R. Wessäly, *Transparent optical network design with sparse wavelength conversion*, ZIB-Report 02–34, October 2002.

[11] R. Hülsermann, M. Jäger, A. Koster, S. Orlowski, R. Wessäly and A. Zymolka, *Availability and Cost Based Evaluation of Demandwise Shared Protection*, in ITG Workshop on Photonic Networks, VDE Verlag 2006.

[12] S. Salsano, A. Botta, P. Iovanna, M. Intermite and A. Polidoro, *Traffic engineering with OSPF-TE and RSVP-TE: Flooding reduction techniques and evaluation of processing cost*, Computer Communications Volume 29, Issue 11, July 2006, Pages 2034-2045.

[13] S. Ioannidis, G. Apostolopoulos, K. Anagnostakis, N. Nikiforakis, A. Makridakis and C. Gkikas. *Resilience of communication networks: Resilience features of IPv6, DNSSec and MPLS*, Technical report, ENISA 2009.

[14] S.Triukose, Z. Al-Qudah and M. Rabinovich, *Content Delivery Networks: Protection or Threat*, EECS Department, Case Western Reserve University, 2009.

[15] D. Wessels and K. Claffy, *Internet Cache Protocol (ICP) version 2*, Internet Engineering Task Force, RFC 2186, September 1997.

[16] N. Kammenhuber, A. Fessi und G. Carle, *Resilience: Widerstandsfähigkeit des Internets gegen Störungen - Stand der Forschung und Entwicklung*, Technische Universität München, Januar 2010.

[17] E. Rosen, D. Tappan, G. Fedorkow and others, *MPLS Label Stack Encoding*, Network Working Group, RFC 3032, January 2001.

[18] M. Handley, *Why the Internet only just works*, BT Technology Journal, Vol 24 No 3, July 2006.

[19] A. Jung, *SCTP for beginners*. http://tdrwww.exp-math.uni-essen.de/inhalt/forschung/sctp_fb/, eingesehen am 22.03.2010.

[20] E. Athanasopoulos, K. Anagnostakis and E. Markatos, *Misusing Unstructured P2P Systems to Perform DoS Attacks: The Network That Never Forgets*, Foundation for Research & Technology Hellas (FORTH), 2006.

[21] U. Bischof, *Strukturierte P2P Netze*, BTU Cottbus- Seminar P2P Networking, 2005.

[22] Welt Online, *Deutsche AOL-Kunden wegen Stromausfall stundenlang offline*, Artikel 444174, 7 April 2001.

[23] V. Jacobson, *Congestion avoidance and control*, ACM SIGCOMM, Stanford, 1988.

Transition to IPv6

Thomas Jakab
Betreuer: Andreas Müller
Seminar Future Internet SS2010
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: jakab@in.tum.de

KURZFASSUNG

Das Internet Protokoll IPv4 ist die Grundlage des heutigen Internets. Allerdings existiert es schon seit den 80er-Jahren und zeigt heutzutage eine Vielzahl an Schwachstellen, die für eine Ablösung drängen. Das größte Problem ist die zunehmende Adressenknappheit, die eine schnelle Ablösung von IPv4 erzwingt. Dafür wurde das „Internet Protocol version 6“ (IPv6) entwickelt, das auf IPv4 aufbaut und den Adressraum vergrößert. Zur Einführung von IPv6 werden verschiedene Techniken benötigt, die eine Koexistenz und Interoperabilität der beiden Protokolle ermöglichen, da nur eine schrittweise Einführung von IPv6 möglich ist. Der Schwerpunkt dieser Arbeit liegt auf der Vorstellung dieser Techniken. Desweiteren wird gezeigt, weshalb die Einführung von IPv6 noch immer sehr schleppend voranläuft, obwohl alle dafür gebrauchten Technologien gut funktionieren und eine Ablösung sinnvoll und von Tag zu Tag nötiger wird.

SCHLÜSSELWORTE

IPv6, Dual Stack Architecture, Tunneling, 6to4, Teredo, 6rd, Translation, SIIT, TRT

1. EINLEITUNG

Mithilfe von IPv6 kann das Problem des versiegenden IPv4-Adresspools gelöst werden. Die meisten Menschen, die mit Informatik zu tun haben, haben schon von IPv6 gehört, die wenigsten aber damit zu tun gehabt. Deshalb wird diese Thematik in diesem Paper genauer erläutert. Zunächst wird aufgezeigt, weshalb IPv4 in Zukunft nicht mehr benutzt werden kann/soll, danach die für das Verständnis wichtigsten Punkte von IPv6 vorgestellt, wie etwa der Aufbau des Headers. Tiefer ins Detail gehende Information werden hier nicht erläutert, aber referenziert. Der Hauptteil der Arbeit beschäftigt sich mit den verschiedenen Einführungsmethoden von IPv6 sowie den Problemen, die eine großflächige Einführung bis heute verhindert haben. Die Einführungsmethoden lassen sich dabei in drei verschiedene, grundlegende Techniken, nämlich Dual Stack Architecture, Tunneling und Translation, unterscheiden. Jede diese Techniken ist notwendig, um die Einführung von IPv6 problemlos zu ermöglichen, da jede bei unterschiedlichen Szenarien angewandt werden muss. Tunneling wird z.B. verwendet, wenn zwei IPv6 Knoten miteinander (teilweise) über ein IPv4-Netz kommunizieren. Translation hingegen wird benötigt, wenn ein reiner IPv4-Knoten Pakete an einen reinen IPv6-Knoten sendet. Bei diesem Fall bietet Tunneling keine Lösung. Trotz dieser Vielfältigen Möglichkeiten, IPv6 problemlos einzuführen wird das neue Protokoll noch so gut wie gar nicht verwendet. Im letzten Teil der Arbeit wird auf die Gründe dafür, wie z.B. das ständige Verbessern und Anpassen von IPv4 an neue Bedürfnisse, eingegangen und Stellung zu diesem Problem genommen.

2. ABLÖSUNGSGRÜNDE FÜR IPV4

IPv4, standardisiert in [2], wurde Anfang der 80er Jahre für eine geringe Anzahl von Nutzern wie Universitäten oder das Militär entwickelt. Damals konnte das rasante Wachstum des Internets nicht vorhergesehen werden. Aus diesem Grund besitzt IPv4 mehrere Schwachstellen, die seine Tauglichkeit als Protokoll für das heutige Internet einschränken:

Das größte Problem IPv4s ist die Anzahl der zu vergebenden Adressen, die theoretisch 2^{32} = 4.294.967.296 beträgt. Dies liegt an dem 32-Bit Format, das IPv4 Adressen verwenden. Eine Adresse besteht aus 4 Blöcken, die jeweils 8 Bit repräsentieren, z.B. 20.135.255.3. Die Spannweite der erlaubten Dezimalzahlen pro Block liegt daher zwischen 0 und 255. Dieser Adressraum von ca. 4 Milliarden Adressen ist mittlerweile fast vollkommen ausgeschöpft. Mithilfe mathematischer Schätzverfahren wird die Erschöpfung des IPv4-Adresspools auf ca. Ende August 2012 datiert[1].

Desweiteren hat IPv4 einige Sicherheitslücken, so ist die Verwendung von IPSec[7], einem Sicherheitsprotokoll zur Authentifizierung und Verschlüsselung, nur optional. Ein weiterer Punkt ist, dass das in Zukunft immer wichtiger werdende Mobile IP, das mobilen Geräten (auch mit statischen IP-Adressen) den Wechsel zwischen Rechnernetzen ermöglicht, sehr schlecht unterstützt wird.

Der Aufbau des IPv4-Headers führt, wie in Kapitel 3.2.3 beschrieben, zu wenig performantem Routing, genauso wie die Implementierung von Fragmentierung(dem Zerlegen eines Paketes in mehrere kleinere).

3. IPV6

Bereits Mitte der 90er Jahre wurden die oben genannten Schwächen von IPv4 bekannt und man begann an einem Nachfolger zu arbeiten: IPv6 (gelegentlich auch IPnG = IP next Generation).1998 wurde IPv6 als neuer Internet Protokoll Standard festgelegt[3]

Beim Design wurde das Hauptaugenmerk auf die Beseitigung der oben genannten Probleme von IPv4 gelegt. So wurde unter Anderem die Adressenlänge auf 128 Bit vergrößert, der Header Aufbau deutlich vereinfacht, die Sicherheitsfunktionen verbessert und die Autokonfiguration von Knoten eingeführt.

3.1 IPv6 Adressen

Die wichtigste Änderung in IPv6 ist das neue Adressenformat:
Zur Behebung der Knappheit von IPv4-Adressen wurden IPv6-Adressen auf 128 Bit vergrößert. Dadurch wird die Anzahl der theoretisch möglichen Adressen auf 2^{128} vergrößert, dies entspricht ca. $3{,}4*10^{38}$, eine Zahl, die das Problem der Adressenknappheit voraussichtlich für immer lösen wird. Veranschaulichen lässt sich diese Menge an Adressen mit folgendem Beispiel: „Je Quadratmillimeter der Erdoberfläche stellt IPv6 667 Billiarden Adressen bereit.“[9] Um diesen 4mal längeren Adressentyp übersichtlich darstellen zu können, werden

IPv6 –Adressen mithilfe von Hexadezimalzahlen dargestellt, die zu 8 Vierergruppen angeordnet werden(siehe Tabelle 2). Jede dieser Gruppen repräsentiert 16 Bit und wird mit einem Doppelpunkt von den anderen Gruppen getrennt. Desweiteren existieren mehrere Vereinfachungsregeln, um die immer noch sehr langen Adressen zu verkürzen. Tabelle 2 gibt einen Überblick über diese Regeln samt Beispiel.

Tabelle 1. IPv6-Adressen und Vereinfachungsregeln

Regel	**Adresse**
BeispielPv6-Adresse	2001:0db8:85a3:0000:0000:8a2e:0000:7344
Führende Nullen dürfen weggelassen werden	2001:db8:85a3:**0**:**0**:8a2e:0:7344 (dabei muss aber jede Gruppe aus min. einer Zahl bestehen)
Aufeinander folgende Nullerblöcke dürfen einmal weggelassen werden	2001:db8:85a3**::**8a2e:0:7344 (dabei werden die weggelassenen Gruppen durch 2 Doppelpunkte repräsentiert)

Ähnlich zu IPv4 ist eine IPv6 Adresse auch in einen globalen und einen lokalen Teil aufgegliedert. Der globale Teil (Netzwerkteil) ist weltweit einzigartig und verweist auf ein Netzwerk, während der lokale Teil (Interface Identifier/Geräteteil) nur innerhalb dieses Netzwerkes eindeutig sein muss und z.B. ein Interface eines PCs identifiziert. Allerdings wird die Aufteilung nicht mehr mithilfe der Subnetzmaske vorgenommen, sondern einfach die Präfixlänge, also die Länge des Netzwerkteils, in Bits angegeben, z.B. würden bei folgender Adresse „2001:db8:85a3::8a2e:0:7344/32“ die ersten 32 Bit das Netzwerk identifizieren, der globale Teil also „2001:db8::/32 “, der lokale Teil „85a3::8a2e:0:7344“ lauten.

Die aus IPv4 bekannten **Broadcast**-Adressen, mit deren Hilfe alle Interfaces in einem Zielnetz angesprochen werden, existieren in IPv6 nicht mehr, da durch Broadcasts häufig Interfaces angesprochen werden, die die Nachricht (IP-Paket) nicht benötigen und deshalb verwerfen. Gebraucht wurden diese in IPv4 für das **„Address Resolution Protocol“** (ARP), mit dessen Hilfe in IPv4 die Link-Layer-Address zu einer IP-Adresse ermittelt wird. Statt dieser nicht performanten Nachrichtenübermittlung mit Broadcasts über ARP wird in IPv6 das **„Neighbour Discovery Protocol“** (NPD) benutzt. Dieses verwendet **Multicasts**, bei denen eine Nachricht an die Gruppe der Interfaces geschickt wird, die an dieser Multicast-Adresse registriert sind. Eine weitere Neuerung ist die Verwendung von **Anycast**-Adressen, an denen ebenfalls jeweils eine Gruppe von Interfaces registriert ist. Wird eine Nachricht an eine Anycast-Adresse geschickt, wird diese jedoch nur an ein Mitglied der Gruppe weitergeleitet. Die Auswahl des Empfängers wird von den Routern anhand der Routing-Protokolle getroffen. Multi- sowie Anycast-Adressen können nur als Empfänger und nie als Absender auftreten. Dieser muss immer eine **Unicast**-Adresse, die einem Interface zugeordnet ist, sein. Natürlich können Pakete auch an Unicast-Adressen gesendet werden, wobei ein PC mehrere Interfaces und somit auch mehrere Adressen haben kann.

IPv6 bietet außerdem die Möglichkeit der Autokonfiguration von Knoten, mit deren Hilfe Knoten anhand ihrer physikalischen MAC-Adresse selbst internetfähige IP-Adressen ableiten können und so automatisch eine Netzwerkverbindung aufbauen können.

Während der Entwicklung von IPv6 wurde diskutiert, ob die neuen Adressen aus 64 oder 128 Bit bestehen sollen. Einen Ausschlag für die größere Variante gab die Möglichkeit nun wieder „verschwenderisch“ mit den Adressen umgehen zu können wie zu Anfang der IPv4 Zeiten; d.h. man kann bestimmte Präfixe für besondere Zwecke reservieren. Einige dieser Präfixe sind für Übergangsmechanismen von IPv4 zu IPv6 reserviert und werden in Kapitel 4 vorgestellt. Eine komplette Liste aller Eigenschaften von IPv6-Adressen findet sich in [10] und [11].

3.2 IPv6 Header

Ein IP Paket besteht immer aus einem Header, in dem für die Übertragung wichtige Daten gespeichert werden, sowie der Payload, also den zu verschickenden Daten, die an den Header angehängt werden. Für IPv6 wurde der IPv4-Header überarbeitet und deutlich vereinfacht. Im Folgenden wird der IPv6-Header vorgestellt und seine Vorteile gegenüber der IPv4 Version werden erläutert.

3.2.1 IPv6 Header Aufbau

Der Header hat eine fixe Länge von 320 Bit, davon entfallen 256 auf Ziel- und Quelladresse. Die genaue Aufteilung ist Abbildung 1 zu entnehmen, wobei die Zahl in Klammern die Anzahl an Bits repräsentiert.

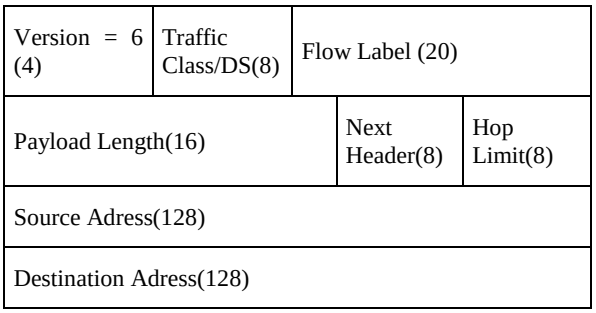


Abbildung 1. Aufbau des IPv6-Headers

Das **„Version“**-Feld ist immer mit der 6 besetzt und gibt an, dass es sich um ein IPv6 Paket handelt.

Das Feld **„Traffic Class/DS“** erlaubt es, Pakete beim Routing differenziert zu behandeln. Werte in diesem Feld werden von Anwendungen und Netzüberwachungsinstanzen gesetzt. Dieses Feld wurde aus dem IPv4 Header übernommen. Eine genaue Definition findet sich in [4].

Mithilfe des neu eingeführten **„Flow Label“** lassen sich Datenströme markieren und können so bevorzugt weitergeleitet werden, wodurch Streaming z.B. für Videokonferenzen deutlich effizienter funktioniert.

Die **„Payload Length“** gibt die Länge des Paketes abzüglich der Header-Länge an, also die Länge der „Nutzlast“. Die Länge eventuell verwendeter Erweiterungsheader(siehe 3.1.2) wird mitgezählt.

„Next Header“ enthält den Code, welcher den Typ des als nächstes folgenden Headers angibt. Dieser kann zum Beispiel ein Erweiterungsheader oder aber auch ein UDP oder TCP Header sein.

„Hop Limit“ bezeichnet die maximale Anzahl an Routern, die ein Paket durchlaufen darf, bis es sein Ziel erreicht. Pro durchlaufenen Router wird der Wert um 1 reduziert. Sollte der

Wert 0 werden und das Paket noch nicht am Ziel sein, wird es verworfen und eine Fehlernachricht an die **„Source Adress“** geschickt. Da es sich um ein 8 Bit Feld handelt, beträgt der maximale „Hop Wert“ 255.

3.2.2 Erweiterungs-Header

Ein zentrales Konzept in IPv6 sind die sogenannten „Erweiterungs-Header“. Mit ihrer Hilfe lassen sich zusätzliche Routing-Optionen, wie z.B. die Fragmentierung von Paketen oder Authentisierung leicht implementieren. Jeder Header-Typ besitzt einen eindeutigen Code. Wird ein Erweiterungs-Header verwendet, muss sein Code in dem „Next Header“-Feld des Basis-Headers eingetragen werden. Werden mehrere Erweiterungs-Header benutzt, dann steht der Code des als 2tes verwendeten Headers im „Next Header“-Feld des an erster Stelle stehenden usw. Dieses Konzept bietet zwei zentrale Vorteile:

Zum Einen müssen Router bei der Weiterleitung nicht alle Header auswerten, sondern können anhand der Codes die für sich Relevanten identifizieren. Dadurch läuft das Routing wesentlich effizienter ab als in IPv4, wo alle Router z.B. das „Options and optional Padding“-Feld auswerten müssen.

Zum Anderen können neue Option einfach mithilfe neuer Erweiterung-Header implementiert werden, wodurch IPv6 sehr flexibel und erweiterbar ist.

Die Reihenfolge, in der die verschiedenen Header verwendet werden, ist wie in Tabelle 2 festgelegt, um die Auswertung beim Routing weiter zu vereinfachen, aber auch aus Sicherheits-gründen. So müssen z.B. geheim zuhaltende Daten nach dem Verschlüsselungs-Header stehen, fürs Routing wichtige allerdings davor, da sie sonst eben verschlüsselt sind.

Tabelle 2. IPv6-Header Reihenfolge

Stelle	Header-Typ	Code
1	IPv6-Basis Header	-
2	Hop-by-Hop Options Header	0
3	Destination Options Header	60
4	Routing Header	43
5	Fragment Header	44
6	Authentication Header	51
7	Encapsulating Security Payload Header	50
8	Destination Options Header	60
9	Upper Layer Header	z.B. TCP = 6

Jeder Erweiterungs-Header besteht aus dem „Next-Header“-Feld und dem „Header Extension Length“-Feld sowie je nach Typ über weitere Felder. Die wichtigsten Header werden im Folgenden kurz beschrieben, eine ausführliche Beschreibung der Erweiterungs-Header findet sich in [5].

„Hop-by-Hop Options“ muss als einziger Erweiterungs-Header von jedem Router, den das Paket durchläuft, ausgewertet werden. Mit seiner Hilfe können Sonderbehandlungen des Paketes durch die Router festgelegt werden.

Der **„Destination Options Header“** kann als einziger Erweiterungs-Header zweimal vorkommen. Er enthält optionale Informationen, wie z.B. die Möglichkeit, extra große Daten zu verschicken. Der Header, der an dritter Stelle (siehe Tabelle 2) steht, wird von allen Knoten ausgewertet, die das Paket durchlaufen muss, der an achter Stelle nur vom Zielknoten.

Der **„Routing Header“** enthält eine Liste von Routern, die das Paket durchlaufen muss. Nur die in der Liste stehenden Router müssen den Header bearbeiten.

Fragmentierung soll in IPv6 aus Perfomance-Gründen vermieden werden. Ist sie dennoch vonnöten, werden die dazu erforderlichen Daten im **„Fragment Header“** gespeichert.

Mithilfe der **„Encapsulating Security Payload“** und **„Authentication“** Header können sichere Datenübertragungen realisiert werden. Mit Ersterem können Daten verschlüsselt werden, mit dem Anderen werden durch Authentisierung Angriffe unterbunden, bei denen sich der Angreifer als Empfänger oder Sender ausgibt. Weiterführende Informationen zu diesem Thema lassen sich in [6] finden.

3.2.3 Vorteile des IPv6 Headers

Um die Vorteile besser darstellen zu können, wird kurz der IPv4 Header vorgestellt. Eine ausführliche Beschreibung liefert [5] bzw. [2].

Version = 4	Header Length	Type of Service	IP Packet Length			
Packet Identification Number (ID)			0	DF	MF	Fragment Offset
Time to Live (TTL)		Protocol	Header Checksum			
Source Adress						
Destination Adress						
Options and optional Padding						

Abbildung 2. Aufbau des IPv4-Headers

Ein Problem von IPv4 ist der komplexe Aufbau des Headers, wodurch das Routing verlangsamt wird, sowie aufgrund des **„Options and optional Padding“**-Feldes die variable Länge des Headers, weshalb das Feld **„Header Length“** benötigt wird. In IPv6 ist mithilfe der Erweiterungs-Header die Header-Länge fix auf 320 Bit festgelegt – das Header Length Feld somit überflüssig.

Die gesamte 2. Zeile in Abbildung 2, die nur der Fragmentierung gilt, wurde ebenfalls entfernt und stattdessen der „Fragment Header“(siehe 3.2.2) eingeführt. Des Weiteren wurde das Fragmentierungs-Konzept aus Performancegründen in IPv6 stark überarbeitet: so kann nun, anders als in IPv4, nur noch der „Quell-Router“ Pakete fragmentieren, wodurch das Routing effizienter abläuft.

Das Feld **„Header Checksum“** wurde ebenfalls gestrichen, da „die Fehlererkennung der heutzutage verwendeten Link-Layer Protokolle so zuverlässig ist, dass man in der Netz-Schicht auf eine nochmalige Fehlersicherung durch Prüfsummen verzichten

kann“[8]. Die dadurch wegfallende Überprüfung der „Checksum“ führt natürlich auch zu einer Steigerung der Performance beim Routen.

4. EINFÜHRUNGSMETHODEN

Da es unmöglich ist das gesamte Internet auf einmal auf IPv6 umzustellen und es somit zu einer langsamen, schrittweisen Einführung kommt, werden die beiden Internetprotokolle für lange Zeit nebeneinander existieren und gezwungenermaßen auch miteinander kommunizieren müssen. So muss ein in einem IPv6-Netz befindlicher Knoten Pakete an ein IPv4-Netzknoten schicken können, oder zwei IPv6-Knoten müssen miteinander über ein IPv4-Netz Daten austauschen können und andersherum. Um diese Kompatibilität herzustellen wurden verschiedene Verfahren entwickelt, die sich in drei Kategorien einteilen lassen und deren Wichtigste im Folgenden vorgestellt werden(siehe auch [12]).

4.1 Dual Stack Architecture

Für Knoten, die häufig IPv4 sowie IPv6 Anfragen beantworten müssen, wie z.B. Server, ist die Implementierung der Dual Stack Architecture sinnvoll. Dank ihr kann sich der Knoten bei IPv4 Anfragen wie ein reiner IPv4 Knoten, bei IPv6 Anfragen wie ein reiner IPv6 Knoten verhalten. Um dies zu realisieren, muss der Knoten über eine IPv4- und eine IPv6-Adresse verfügen.

Sinnvoll ist dafür die Nutzung der **„IPv4-mapped-IPv6-Adresse“**(im Folgenden mit „mapped-Adresse“ abgekürzt). Für diesen Zweck wurde der Präfix „::ffff/96“ reserviert und die Mischschreibweise aus IPv6- und IPv4-Adressen eingeführt, z.B. „::ffff:128.2.33.200“. Natürlich kann diese Adresse auch als reine IPv6-Adresse geschrieben werden, in dem die Dezimalwerte ins Hexadezimalsystem umgerechnet und jeweils zwei Blöcke zusammengefasst werden: „::ffff:8002:21C8“. Mithilfe dieser Adresse können also IPv4-Adressen automatisch aus IPv6-Adressen abgeleitet werden und andersherum.

Diese Adressen werden benötigt, wenn z.B. ein IPv4-Client einen IPv6-Dienst an einem Dual Stack Server anfrägt. In diesem Fall muss seine IPv4-Adresse auf eine IPv6-Adresse abgebildet werden:

Empfängt ein Knoten mit Dual Stack Implementierung ein IPv4 Paket, bildet es die IPv4-Adresse auf eine mapped-Adresse ab und leitet es an die höher gelegene Protokollschicht weiter. Falls der Knoten eine Antwort an den Sender zurückschickt, erkennt er die mapped-Adresse und wandelt sie wieder in die IPv4-Adresse um.

Bei Empfang eines IPv6 Pakets verhält sich der Knoten wie ein reiner IPv6 Knoten.

Außerdem lassen sich die einzelnen Funktionen deaktivieren, so dass ein Dual Stack auch als reiner IPv4 bzw. IPv6 Knoten betrieben werden kann.

4.2 Tunneling

Es kann vorkommen, dass obwohl zwei IPv6-Knoten miteinander kommunizieren, ein IPv6-Paket über ein reines IPv4 Netz geroutet wird. In diesem Fall muss es in ein IPv4-Paket umgewandelt werden um ans Ziel zu kommen. Eine Möglichkeit dafür ist das „Tunneling“. Dabei wird das IPv6-Paket in ein IPv4-Paket eingewickelt und so über den „Tunnel“, das IPv4-Netz, übertragen. Wenn der Tunnel überbrückt wurde und das Paket sich wieder im IPv6-Netz befindet, wird der IPv4-Rahmen entfernt und das IPv6-Paket weitergeleitet. Dieser Vorgang geschieht für den Benutzer transparent, da beim Einpacken des Pakets das Hop-Limit Feld des IPv6-Headers um 1 verringert wird. Erst beim Weiterleiten nach dem Entpacken am Tunnelendpunkt wird das Hop-Limit wieder um 1 dekrementiert. Der Tunnel über beliebig viele IPv4-Knoten wird also als nur ein „Hop“ gezählt. Nachteile dieses Verfahrens sind der Performanceverlust durch das Ein- und Auspacken sowie die geringer werdende Kapazität für die Nutzlast, da der Overhead der Pakete vergrößert wird, wodurch es im ungünstigsten Fall zu Fragmentierung und einem weiteren Performanceverlust kommen kann. „Tunneling“ kann entweder manuell oder automatisch geschehen. Bei der manuellen Variante müssen der Tunnelstart- und Tunnelendpunkt fest konfiguriert werden, wodurch ein hoher Aufwand entsteht. Allerdings kann dies aus Sicherheitsaspekten sinnvoll sein. Als Endpunkt kann eine Anycast-Adresse benutzt werden, wodurch von mehreren Routern nur einer zum Empfang gewählt wird. Dadurch erhöht sich die Ausfallsicherheit. Um automatisches „Tunneling“ zu implementieren gibt es verschiedene Varianten, von denen die wichtigsten in diesem Abschnitt erläutert werden.

4.2.1 6to4

6to4 ist die weltweit am häufigsten benutzte Technik um IPv6 Konnektivität herzustellen[13]. Um 6to4 nutzen zu können, muss eine global eindeutige IPv4-Adresse vorhanden sein. Die Kommunikation erfolgt über spezielle 6to4-Router.

Für 6to4 wurde ähnlich zu „IPv4-mapped-IPv6“ ein Adress-Präfix reserviert. Dieser hat die Form „2002::/16“. Der globale Teil einer 6to4-Adresse setzt sich aus diesem Präfix und der Hexadezimalrepräsentation der IPv4-Adresse des 6to4 Routers zusammen; z.B. „2002:8002:21C8::/48“ bei der IPv4-Adresse „128.2.33.200“. Dadurch wird jeder IPv4 Adresse und damit auch jedem 6to4 Router ein eigenes /48 Netz zugeordnet.

Innerhalb eines 6to4-Netzes können Knoten problemlos miteinander kommunizieren(siehe Host A und Host B in Abbildung 3). Kommt es zu einer Kommunikation mit einem Knoten in einem anderen 6to4-Netz(Host A mit Host C), packt der sendende 6to4 Router(R1) das IPv6-Paket in ein IPv4 Paket ein und sendet es an den empfangenden 6to4 Router(R2). Die IPv4-Adresse des empfangenden Routers kann er dabei aus der IPv6-Adresse des Adressaten(Host C) ableiten. Zusätzlich wird die Protokollnummer im Header des IPv4-Paketes auf 41 gesetzt, woran der empfangende Router erkennt, dass es sich um ein getunneltes IPv6-Paket handelt, entpackt es und leitet es weiter. Auch hier kann an eine Anycast-Adresse gesendet werden, wodurch der nächstgelegene 6to4 Router ausgewählt wird und nicht ein Bestimmter adressiert werden muss. Sie ist in [14] festgelegt und lautet „2002:c058:6301::“.

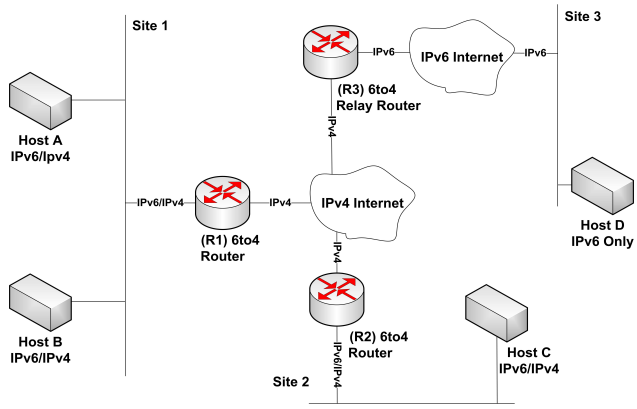


Abbildung 3. Möglichkeiten der Kommunikation von 6to4 Komponenten. In Anlehnung an [15]

Wird ein Knoten in einem reinen IPv6-Netz adressiert(Host A an Host D), muss ein 6to4 Relay Router dazwischengeschaltet

werden, da die IP-Adresse des Empfängers(Host D) nicht mehr wie vorher aus der Adresse des Routers abgeleitet wird.
Der Nachteil dieser Technik ist die Notwendigkeit der Benutzung spezieller Router, und dass 6to4 nicht kompatibel zu NAT (Network Address Translation) ist, da unter anderem die Protokollnummer 41 von NATs standardmäßig nicht verarbeitet werden kann und bei der Verwendung von NATs nicht immer eine global eindeutige IPv4-Adresse existiert.[16]

4.2.2 Teredo

Um bei der Verwendung von NAT nicht auf „Tunneling“ verzichten zu müssen entwickelte Microsoft Teredo[17]. Teredo ist als Übergangslösung bestimmt, bis 6to4 auch mit NAT funktioniert. Da Teredo einen großen Overhead erzeugt, nicht so performant ist wie andere Techniken und Sicherheitsrisiken aufweist, ist, falls möglich, eine andere Art zu tunneln zu verwenden.

Bei Teredo werden die IPv6-Pakete als UDP-Nutzlast in einem IPv4-Paket übertragen. Sie werden also in UDP-Pakete anstatt in IPv4-Pakete eingekapselt, damit der NAT Paketfilter sie nicht zurückweist.

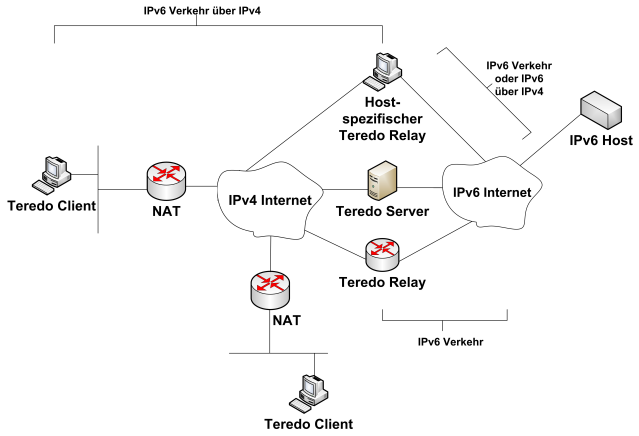


Abbildung 4. Teredo Komponenten und Zusammenspiel. In Anlehnung an [18]

Der **Teredo Client** (siehe Abbildung 4**)** ist ein IPv4-Knoten, der Zugang zu IPv6 Internet will. Dies geschieht mithilfe des **Teredo Servers** und des **Teredo Relays**. Der Server hat mittels einer globalen IPv4-Adresse Zugang zum IPv4-Internet und kann so dem Client eine spezielle IPv6-Adresse zuordnen, die **Teredo Address**. Sie setzt sich aus dem Teredopräfix „2001::/32“, der IPv4-Adresse des Servers und der des Clients sowie weiteren Feldern zusammen. Das **Teredo Relay** ist ein Router, der IPv6-Pakete an die Clients weiterleiten kann. Ein **Hostspezifischer-Teredo-Relay** kann über IPv4 und IPv6 kommunizieren und somit über IPv4 direkt mit dem Client in Verbindung treten – er benötigt deshalb kein zwischengeschaltetes Teredo-Relay wie der Server.

4.2.3 6rd(6 Rapid Deployment)

Eine relativ junge Technik ist die von Rémi Després entwickelte und im Januar 2010 in [19] veröffentlichte 6rd Tunnelmethode, die auf 6to4 aufbaut, aber versucht deren Schwächen zu beseitigen: So wird anstatt des festen 6to4 Präfixes ein pro ISP(Internet Service Provider) eindeutiges Präfix genutzt. Der Router des Benutzers kapselt bei Anschluss an ein IPv4-Netz die 6rd-Adresse dann in eine IPv4-Adresse und leitet das Paket an eine dem ISP gehörende Gateway, die das IPv6-Paket wieder entpackt. Dadurch ist sichergestellt, dass der ISP nur zu ihm gehörende Pakete weiterleiten muss, anders als bei 6to4, wo aufgrund des einheitlichen Präfixes nicht kontrolliert werden kann, woher der Traffic kommt. Es sind wie bei 6to4 auch Verbindungen von 6rd-Netz zu 6rd-Netz über ein IPv4 Netz möglich; dazu ist keine extra Gateway von Nöten.

Der globale Teil von 6rd-Adressen besteht aus dem maximal 32 Bit großen 6rd-Präfix des ISP sowie der IPv4-Adresse des Kunden. Der französische ISP „free“ hat als erstes 6rd seinen Kunden angeboten und dafür den Adresspräfix „2a01:0e00:::/26” erhalten[20].

4.3 Translation

Eine weitere Möglichkeit der Interoperabilität von IPv4 und IPv6 wird durch Übersetzen der Protokolle erreicht. Diese wird gebraucht, wenn Knoten, die sich in einer reinen IPv4-Umgebung befinden mit Knoten kommunizieren müssen, die in einer reinen IPv6-Umgebung stehen oder andersherum. Dann ist nämlich kein „Tunneling“ möglich. Zwei Beispiele dafür sind die hier vorgestellten SIIT(Stateless IP/ICMP Translation) und TRT (Transport Relay Translator).

4.3.1 SIIT

Bei SIIT-Kommunikation werden ausgehende IPv6-Pakete von einem speziellen SIIT-Router, über den der IPv6 Knoten mit dem IPv4-Netz kommuniziert, in IPv4, sowie eingehende Pakete aus dem IPv4-Netz in IPv6 übersetzt. Dazu wird eine spezielle IPv6-Adresse benötigt, die „**IPv4-translated IPv6-Address**“. Sie ist aus dem Präfix „0:0:0:0:FFFF::/96“ und einer IPv4-Adresse zusammengesetzt. Diese Adresse muss für die Dauer der Kommunikation dem IPv6-Knoten fest zugeteilt werden.

Die Felder der Header lassen sich ohne Schwierigkeiten automatisch erzeugen bzw. werden verworfen(wie die IPv4 Optionen), allerdings lassen sich viele IPv6-Erweiterungsheader nicht übersetzen(z.B. der Authentification-Header). Des Weiteren müssen noch zusätzlich Protokolle, wie ICMP, übersetzt werden. Eine genaue Beschreibung der Übersetzungsvorgänge findet sich in [21].

4.3.2 TRT

TRT wird eingesetzt um auf IPv6 basierende TCP/UDP-Anwendungen mit auf IPv4 basierenden zu verbinden. Dies funktioniert nur bei bidirektionaler Kommunikation unter Verwendung eines besonderen DNS-Servers.

Zum Aufbau der Kommunikation bekommt der IPv6-Knoten(C6 in Abbildung 5) über den DNS-Server eine von diesem speziell erstellte IPv6-Adresse, deren letzte 32 Bit aus der IPv4-Adresse des IPv4-Knotens(S4) bestehen, übermittelt. Die Kommunikation der beiden Knoten läuft nun über das TRT, welches aus den IPv6-Paketen IPv4-Pakete macht und an die Adresse des Knotens(S4) schickt. In die andere Richtung funktioniert die Kommunikation nach demselben Prinzip. TRT ist in [22] spezifiziert.

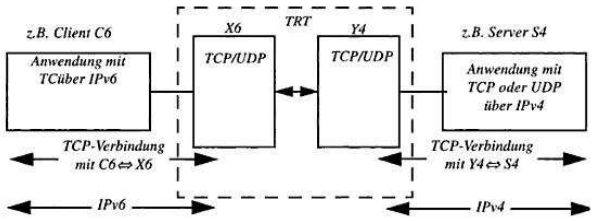


Abbildung 5. Funktionsweise von TRT[23]

5. STELLUNGNAHME ZU EINFÜHRUNGSPROBLEMEN

Trotz der in Kapitel 3 genannten Vorteile von IPv6, seiner schon relativ langen Existenz sowie der zunehmenden Knappheit von IPv4-Adressen wird das neue Protokoll noch so gut wie gar nicht genutzt. Laut eines Reports von Google[13] von 2008 haben gerade einmal 0,238% der Internetnutzer weltweit IPv6 Konnektivität und bevorzugen diese auch. Ein Grund dafür dürfte die typische Mentalität „wozu soll ich mein System ändern, wenn es doch funktioniert“ sein. Ein Weiterer, dass die meisten Benutzer wenig Ahnung haben, wie das Internet überhaupt funktioniert und sich deshalb keine Gedanken über diese Problematik machen. Desweiteren bieten die meisten ISPs keine IPv6-Konnektivität an, da sie keine Nachfrage verspüren und deshalb die Investitionen scheuen. Außerdem spielt noch eine Rolle, dass IPv4 ständig weiter verbessert wird, um das Ende des Protokolls möglichst lange hinauszuzögern: So wurde z.B. NAT eingeführt, um die Adressenknappheit hinauszuzögern, Sicherheitsbedenken wurden mit der nachträglichen Integration von IPSec umgangen und heutzutage wichtige Funktionen wie Mobilität wurden ebenfalls nachträglich (suboptimal) ermöglicht. Es wird also immer wieder repariert und erweitert, anstatt das neue Protokoll IPv6 zu verwenden, welches all diese Probleme vorerst lösen kann.

Dies führt wohl zu einer Art Teufelskreis, der erst durchbrochen werden kann, wenn die ISPs kurz davor sind, keine IPv4-Adressen mehr zu bekommen. Ob die Umstellung dann mithilfe von schnell zu implementierenden Einführungsmethoden wie 6rd(der französische ISP „free“ hat sein Netz innerhalb von 5 Wochen dank 6rd zu IPv6 erweitert) rechtzeitig zu schaffen ist, bleibt abzuwarten, scheint aber wahrscheinlich. Dennoch befinden sich die ISPs in einer Bringschuld: Umstellen werden sie sowieso irgendwann müssen und solange kein Angebot existiert wird die Nachfrage auch nicht steigen. „free“ und einige Andere sind hier schon mit gutem Beispiel vorangegangen.

6. ZUSAMMENFASSUNG

Bei der Entwicklung von IPv6 wurde darauf geachtet, aus den Problemen von IPv4 zu lernen und ein flexibles, leicht erweiterbares Protokoll zu schaffen. Erreicht wurde dies vor allem durch die enorme Vergrößerung des Adressraums auf 128 Bit und der damit möglichen Klassifizierung von IP-Adressen, der Fähigkeit zur Autokonfiguration und dem Ersetzen von Broadcast durch Multi und Anycast. Desweiteren spielt auch die Verbesserung des Headers eine große Rolle: So wurden Erweiterungs-Header eingeführt, überflüssige Felder entfernt und dadurch das Routing performanter gestaltet.

Da keine „Big-Bang“-Ablösung von IPv4 zu bewerkstelligen ist, wurden verschiedene Einführungsstrategien entwickelt, die sich in die drei Gruppen „Dual Stack Architecture“, „Tunneling“ und „Translation“ einteilen lassen. Jede dieser Gruppen ist für bestimmte Einsatzgebiete gut geeignet, wobei bei den „Tunneling“ Techniken vor allem der neue Ansatz 6rd sehr vielversprechend ist, da die auf 6to4 basierende Methode relativ leicht, schnell und günstig zu implementieren ist und deshalb die bis jetzt sehr langsam stattfindende Einführung von IPv6 beschleunigen könnte. Wichtige „Translation“-Techniken sind SIIT und TRT, bei denen die Protokolle automatisch in das jeweils andere Format übersetzt werden.

Gründe für die schleppende Einführung sind die noch geringe Nachfrage der Nutzer, aber auch das fehlende Anbieten von IPv6 durch die Internet Service Provider. Diese voneinander abhängigen Gründe werden die Einführung jedoch nicht aufhalten können, da bald keine IPv4-Adressen mehr vergeben werden können. Die Techniken und das Know-How sind vorhanden, jetzt muss nur noch gehandelt werden.

7. REFERENZEN

[1] Husten,Geoff, „IPv4 Adress Report“,11 May 2009. http://www.potaroo.net/tools/ipv4

[2] Information Science Institute USC, „Internet Protocol“, RFC 791, IETF, September 1981.

[3] S. Deering, R. Hinden, „IPv6“, rfc 2460, IETF, December 1998.

[4] K. Nichols, S. Blake, F. Baker, D. Black, „Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers“, RFC 2474, IETF, December 1998.

[5] Herbert Wiese, „Das neue Internetprotokoll IPv6“, 2002, Carl Hanser Verlag München Wien, ISBN 3-446-21685-5, Seiten 28-42.

[6] Silvia Hagen, „IPv6 Essentials“, 2006, O’Reilly Media, Inc., ISBN 978-0-596-10058-2, Chapter 2

[7] S. Kent, K. Seo, „Security Architecture for the Internet Protocol“, RFC 4301, IETF, December 2005

[8] Herbert Wiese, „Das neue Internetprotokoll IPv6“, 2002, Carl Hanser Verlag München Wien, ISBN 3-446-21685-5, Seite 27.

[9] Hans Peter Dittler, „IPv6-das neue Internet-Protokoll“, 1998, dpunkt-Verl., ISBN 3-932588-18-5,Seite 36 a

[10] Hans Peter Dittler, „IPv6-das neue Internet-Protokoll“, 1998, dpunkt-Verl., ISBN 3-932588-18-5,Kapitel 3

[11] R. Hinden, S. Deering, „IP Version 6 Addressing Architecture“,RFC 4291, IETF, February 2006

[12] R. Gilligan, E. Nordmark, „ Transition Mechanisms for IPv6 Hosts and Routers“, RFC 2893, IETF, August 2000

[13] Steinar H. Gunderson, „Global IPv6 statistics- Measuring the current state of IPv6 for ordinary users“,RIPE Meeting 57, October 2008

[14] C. Huitema, „An Anycast Prefix for 6to4 Relay Routers“, RFC 3068,IETF,June 2001

[15] Silvia Hagen, „IPv6 Essentials“, 2006, O’Reilly Media, Inc., ISBN 978-0-596-10058-2, Page 265

[16] B. Carpenter, K. Moore, „Connection of IPv6 Domains via IPv4 Clouds“, RFC 3056,IETF February 2001

[17] C. Huitema, „Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)“, RFC 4380,IETF, February 2006

[18] Microsoft TechNet, „Überblick zu Teredo“, 05.März.2004, http://www.microsoft.com/germany/technet/datenbank/articles/600330.mspx

[19] R. Despres, „IPv6 Rapid Deployment on IPv4 Infrastructures (6rd)“, RFC 5569,IETF, January 2010

[20] A. Classen, „IPv6@free-native IPv6 to the User“, RIPE Meeting 58, 05.05.2009

[21] E. Nordmark, „Stateless IP/ICMP Translation Algorithm (SIIT)“, RFC 2765,IETF, February 2000

[22] J. Hagino, K. Yamamoto, „ An IPv6-to-IPv4 Transport Relay Translator“, RFC 3142,IETF, June 2001

[23] Herbert Wiese, „Das neue Internetprotokoll IPv6“, 2002, Carl Hanser Verlag München Wien, ISBN 3-446-21685-5, Seite 298

Stream Control Transmission Protocol (SCTP)

Violin Yanev
Betreuer: Nils Kammenhuber
Seminar Future Internet SS2010
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: yanev@in.tum.de

KURZFASSUNG

Das Transportprotokoll SCTP wurde entwickelt, um Funktionalität anzubieten, welche die weitverbreiteten TCP und UDP nicht bieten. Dieses Protokoll weist starke Ähnlichkeiten zu TCP auf, bietet aber weitere Möglichkeiten, wie die Aufteilung des Bytestroms in einzelne Nachrichten, die Aufspaltung des Datentransfers in sogenannte Streams sowie die verbesserte Ausfallsicherheit und Angriffschutz. Dieses Paper versucht, die zusätzlichen Features, die SCTP implementiert, in einem objektiven Vergleich mit anderen Transportprotokollen zu untersuchen.

Schlüsselworte

SCTP, Transportprotokoll, Multihoming, Multistreaming, Network Resilience, TCP

1. MOTIVATION

TCP ist das zurzeit am meisten verbreitete Transport-Protokoll. Obwohl es eine erhebliche Rolle im Bereich des verlässlichen Datentransfers leistet und geleistet hat, benötigt es gewisse Eigenschaften, die das moderne Internet braucht. Aus diesem Grunde haben einzelne Nutzer von TCP versucht, durch Extensions oder sogar mittels eigener Protokolle ihre Bedürfnisse zu befriedigen.

Die Entwickler von SCTP haben versucht, anhand der Erfahrung mit TCP ein neues Transportprotokoll zu entwerfen. Obwohl es ursprünglich zum Datentransfer innerhalb von PSTN (Public Switched Telephone Network) entwickelt wurde, können auch andere Anwendungen Gebrauch von den SCTP-Features machen.

2. EINFÜHRUNG IN SCTP

Bevor die Vor- und Nachteile von SCTP gegenüber TCP in den folgenden Kapiteln untersucht werden, bietet sich hier eine kurze Einführung in die Konzepte beider Protokolle. Diese hat den Zweck, dem Leser einen Überblick über SCTP zu verschaffen sowie die im Folgenden verwendeten Begriffe zu erklären.

2.1 Begriffserklärung und Annotation

An dieser Stelle muss erklärt werden, welche Bedeutung der Begriff *Stream* in diesem Paper trägt. TCP definiert Stream als eine Bytefolge, die in Pakete aufgeteilt und transportiert wird. SCTP dagegen spaltet den gesamten Datentransfer in mehrere logische Datenströme auf, die auch als Streams bezeichnet werden. In diesem Paper wird der TCP-Begriff als *Bytestream* und der SCTP-Begriff einfach als *Stream* bezeichnet, um Verwirrung zu vermeiden.

Eine SCTP-Verbindung wird auch *Assoziation* genannt, weil ein oder beide Endpunkte mehrere Netzwerkadressen besitzen können, womit der Term „Verbindung“ nicht mehr einen eindeutigen physischen Pfad zwischen den Peers definiert.

Mit *Overhead* ist der Datenzuschlag gemeint, der durch sämtliche Header zusätzlich zu den Benutzerdaten entsteht.

ULP ist eine Abkürzung für Upper-Layer Protocol, was im Fall von SCTP die Anwendung ist.

2.2 Wer kümmert sich um SCTP?

Der Protokoll wurde im Jahr 2000 von der IETF Signaling Transport Workgroup, SIGTRAN, entwickelt. Es wird von der IETF Transport Area Workgroup betreut, unterstützt und aktualisiert. Die Referenz [1] gibt die technische Spezifikation an.

2.3 Eigenschaften von SCTP

SCTP agiert auf der Transportschicht in der Internet Protocol Suite, also zwischen der Internetschicht und der Anwendungsschicht. Es ist verbindungsorientiert wie TCP und hat als kleinste Dateneinheit eine Message (beliebige Folge von Bytes). Die Verbindung wird zwischen genau zwei Endpunkten aufgebaut und kann in beide Richtungen Daten transportieren (full duplex). Jeder Endpunkt kann aber mehrere Netzadressen besitzen, was als „Multihoming“ bezeichnet wird (s. Abschnitt 3.4).

2.4 Paket-Struktur

Das SCTP-Paket (vgl. Abbildung 1) besteht aus einem Common Header und einer beliebigen Folge von Chunks. Der 12 Byte große Header enthält die Source- und Destination Ports (je 2 Byte), den Verification-Tag (4 Byte) zur Validierung der Pakete (s. Abschnitt 3.6) und eine Prüfsumme (4 Byte) als Schutz gegen Störungen auf der Leitung. Die Chunks sind Dateneinheiten, die entweder Nutzdaten oder Kontrollinformation transportieren. Jedes Chunk beginnt mit einem Byte, das den Chunk-Typ angibt (User Data, INIT, ACK, ABORT oder andere Kontrolltypen). Danach folgen Flags (1 Byte), Länge der Chunks und die eigentlichen Daten. Für eine genauere Beschreibung der verschiedenen Chunk-Typen ist der interessierte Leser auf die genaue Spezifikation verwiesen [1].

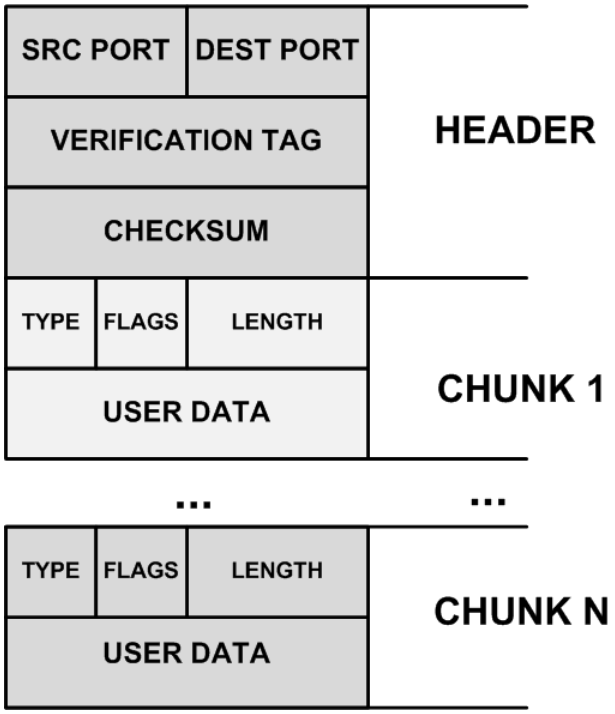


Abbildung 1: Die SCTP-Paketstruktur

3. FEATURES

In diesem Abschnitt werden kurz die Besonderheiten von SCTP dargestellt. Eine nähere Betrachtung und Bewertung der Features findet sich im Abschnitt 4 („Vergleich mit TCP“). Eine Liste findet sich auch in Tabelle 1.

3.1 Verbindungsauf- und abbau

SCTP implementiert einen 4-Way-Handshake-Verbindungsaufbau, der das Cookie-Verfahren verwendet. Das Cookie ist ein Datenblock, der die Verbindungsparameter zum Zeitpunkt des Verbindungsaufbaus enhält und zurück an den Client gesendet wird. Warum es sinnvoll ist, diese Daten im Cookie zu speichern und nicht auf dem Server selbst, wird in Abschnitt 4.3 näher beschrieben.

Feature name	UDP	TCP	SCTP
Connection oriented	No	Yes	Yes
Reliable transport	No	Yes	Yes
Unreliable transport	Yes	No	Yes
Preserve image boundary	Yes	No	Yes
Ordered delivery	No	Yes	Yes
Unordered delivery	Yes	No	Yes
Data checksum	Yes	Yes	Yes
Checksum size (bits)	16	16	32
Partial checksum	No	No	No
Path MTU	No	Yes	Yes
Congestion control	No	Yes	Yes
Multiple streams	No	No	Yes
Multihoming	No	No	Yes
Bundling	No	Yes	Yes

Tabelle 1: Unterstützte Features von UDP, TCP und SCTP

Im ersten Schritt sendet der Client ein INIT Paket (entspricht dem SYN in TCP), worauf der Server mit einem INIT-ACK Paket mit dem darin enthaltenen Cookie reagiert. Im dritten Schritt sendet der Client das erhaltene Cookie in einem COOKIE-ECHO Paket zurück, um die Verbindung zu bestätigen, daraufhin sendet der Server ein COOKIE-ACK Paket, und die Verbindung ist aufgebaut (s. Abbildung 4b). Im dritten und vierten Schritt dürfen bereits Daten übertragen werden, um eine weitere RTT (Round-Trip-Time) als Wartezeit bis zum Senden der ersten Nutzdaten zu vermeiden.

Eine Verbindung wird wie bei TCP per 3-Way-Handshake abgebaut. Dieser Vorgang wird für jeden der beiden Kommunizierenden vollzogen. Wenn beide Teilnehmer keine Daten mehr übertragen möchten, wird die Verbindung getrennt. Dadurch entsteht der Begriff „Half-Open-State“, was bedeutet, dass Daten nur noch in eine Richtung übertragen werden, bis auch der andere Peer mit der Übermittlung fertig ist.

SCTP unterscheidet sich dadurch von TCP, dass eine Assoziation keinen Half-Open-State besitzt, sondern erzwingt, dass beide Teilnehmer die Verbindung aufgeben, sobald alle gepufferten Daten erfolgreich übertragen wurden.

Es ist auch möglich, eine Verbindung mittels ABORT (analog zu RST bei TCP) zu unterbrechen, falls z.B. ein Restart erfolgt oder ein Fehler entdeckt wurde.

3.2 Messages und Zuverlässigkeit der Lieferung

SCTP bietet eine verlässliche Übertragung, d.h. die Ankunft der Daten auf der anderen Seite der Verbindung ist garantiert. Das geschieht ähnlich wie bei TCP: Die Daten werden mit Sequenznummern (SN) versehen, die der Empfänger mit einem SACK Paket (selective acknowledgement) [4] bestätigt. Im Gegensatz zu TCP, das keine Teilung der Daten unternimmt, sondern lediglich eine ununterbrochene Folge von Bytes an den Peer sendet, transportiert SCTP die Daten in Form von Messages (s. Abbildung 2a-b). Diese Messages können in kleinere Stücke fragmentiert werden, wenn sie die ermittelte Path-MTU (Maximum Transmission Unit [5], die Packetgröße, die der Netzwerkpfad erlaubt) überschreiten. Für viele, kleine Messages ist Bündelung vorgesehen, so dass mehrere kleine Nachrichten (jede in einem Data-Chunk verpackt) in einem größeren SCTP-Paket zusammengefasst werden können.

Ein Unterschied zu TCP besteht darin, dass die Sequenz-Nummern bei SCTP pro Message vergeben werden, nicht pro Byte. Dadurch wird z.B. die Berechnung der Datenmenge, die sich auf dem Weg zum Peer befindet, etwas komplizierter, da die Größe der Messages zusätzlich beachtet werden muss.

3.3 Streams in SCTP

Da der Datenstrom in SCTP in einzelnen Nachrichten organisiert ist, besteht die Möglichkeit, die Nachrichten wiederum in einzelne, logisch abgetrennte Ströme aufzuspalten (s. Abbildung 2c).

TCP leistet einen geordneten Transportdienst, d.h. Daten werden in der richtigen Reihenfolge geliefert. Für manche Anwendungen ist das aber nicht für alle Daten oder für gar keine Daten notwendig. Beispielsweise ist die richtige Reihenfolge für Dateien kritisch, nicht aber für Signalnachrichten oder Online-Spiele, wo die Datenübertragung in Echtzeit stattfindet. Um diese Funktionalität zu gewährleisten, erlaubt SCTP mehrere logische Datenströme. Die Reihenfolge der Daten wird für den einzelnen Strom garantiert, und optional kann der Nutzer auch ungeordnete Ströme festlegen. So kann der Nutzer selbst angeben, welche Daten reihenfolgekritisch sind und welche nicht. Das hat auch einen weiteren Vorteil – es verhindert, dass ein Strom, dessen Pakete auf der Strecke verloren gegangen sind, den Fluss der anderen Ströme blockiert. Dieses Problem ist in der Fachwelt als *Head-Of-Line blocking* bekannt [6] und eine vergleichbare Lösung kann in TCP nur mit mehreren Verbindungen verwirklicht werden. Das erschwert aber die Instandhaltung der Verbindung: Die einzelnen Datenflüße sind trotzdem logisch abgespalten, unterliegen nicht ein und derselben Staukontrolle und generieren zusatzlichen Overhead. Außerdem kann eine Anwendung mittels zweier oder mehreren TCP-Verbindungen eine mehrfache Bandbreite beantragen, was die anderen Anwendungen bzw. Benutzer im Netz benachteiligt.

Mit einer Erweiterung [20] von SCTP kann man sogar unzuverläßige Verbindungen definieren, d.h. solche, bei denen verlorene Pakete nicht erneut vermittelt werden, ähnlich wie bei UDP. Es ist sogar möglich, einen unsicheren, aber geordneten Stream zu definieren (diese Funktionalitat bietet UDP nicht). Da SCTP die Daten in Streams einteilt, ist es moglich, unsichere mit sicheren Datenströmen innerhalb einer Assoziation zu multiplexen. Alternativ kann man eine UDP- und eine TCP-Verbindung parallel aufbauen, was aber mit einem zusätzlichen Overhead verbunden ist. Außerdem unterliegt mit SCTP auch die unsichere Verbindung einer Staukontrolle, etwas, was UDP überhaupt nicht beachtet. Diese Erweiterung heißt SCTP-PR (fur Partial Reliability) und bietet eine Reihe weiterer Möglichkeiten. Zum Beispiel kann man spezifizieren, wie lange der Sender warten muss, bevor er nicht bestätigte Pakete verwirft. Die genaue Spezifikation von SCTP-PR ist in [20] enthalten.

SCTP verwaltet die Ströme, indem jedes Data-Chunk mit einer Stream-ID versehen wird. Die maximale Anzahl von Streams wird wahrend der INIT-Phase zwischen Client und Server ausgehandelt.

3.4 Multihoming

SCTP erlaubt, dass einer oder beide Verbindungsteilnehmer mehrere IP-Adressen besitzen können. Daduch entstehen mehrere Möglichkeiten zur Auswahl des Netzwerkpfades zum Peer. Im Falle eines Ausfalls der physischen Verbindung kann der Sender eine alternative Route wählen. Die Verbindung geht nicht verloren, solange es noch mindestens einen aktiven Pfad gibt. Im Allgemeinen bietet die Verbindung dadurch einen besseren Ausfallschutz als eine TCP-Verbindung. Das Multihoming hat eine Einschränkung – es bietet standardmäßig kein Load Sharing [7], es werden also nicht die Kapazitäten aller Netzwerkschnittstellen ausgenutzt. Der Sender wählt einen Pfad und sendet darüber alle Daten, es sei denn der Pfad wird unterbrochen. Erst

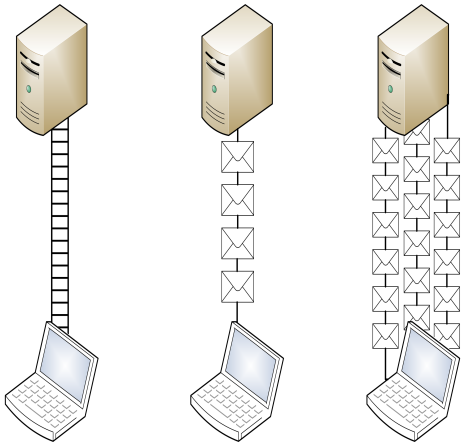


Abbildung 2: Von links nach rechts: (a) TCP mit reinem Bytestrom, (b) Message-orientiertes SCTP, (c) Message-orientiertes SCTP mit Streams

dann versucht SCTP einen anderen Pfad zu finden. Die Pfade, die aktuell nicht zur Datenübertragung verwendet werden, überprüft SCTP laufend auf Verfügbarkeit, indem es HEARTBEAT-Requests an den Peer sendet, der seinerseits mit HEARTBEAT-ACK antwortet. Wenn eine Zeitlang kein ACK ankommt, wird der Pfad als inaktiv gekennzeichnet.

Obwohl mehrere IP-Adressen eingesetzt werden, ist die Portnummer für die bestehende Assoziation eindeutig.

3.5 Stau- und Flusskontrolle

Die Staukontrolle ist ein Mechanismus, der die Überlastung des darunter liegenden Transportnetzes vermeidet und auf spontan enstandene Überlast reagiert (z.B. im Fall einer ausgefallenen Backbone-Verbindung, die den Datentransfer dramatisch verringert). Es basiert auf vier Prinzipien – slow start, congestion avoidance, fast retransmit und fast recovery. Die Staukontrolle ist ein komplexes und von der Implementierung abhängiges Konzept, dessen genauere Betrachtung den Rahmen dieses Papers sprengen würde. SCTP lehnt sich stark an die in TCP definierten [9] Techniken an, der fundamentale Unterschied besteht lediglich im Multihoming-Feature. Referenz [1] legt fest, dass die Staukontrolle pro Empfänger-IP-Adresse durchgeführt werden soll. Nähere Infos befinden sich im Kapitel Congestion control von [1].

Die Flusskontrolle schützt vor Überflutung des Empfängers mit Informationen. Dies kann vorkommen, wenn der Empfänger viel langsamer Daten verarbeitet als der Sender (zum Beispiel bei der Verbindung mit einem PDA). TCP und SCTP ähneln sich auch in der Flusskontrolle, mit der Ausnahme, dass ungeordnete Ströme keine Lücken im Transport aufweisen können, was die Kalkulation des Receiver-Windows beeinflusst [10].

3.6 Paket-Validierung

Das SCTP-Paket enthält im Common-Header einen 32 Bit langen Validierungstag (s. Abbildung 3). Pakete mit falschem Validierungstag werden schweigend verworfen. Dieser Tag hat verschiedene Anwendungen, die wichtigste davon ist ein Schutz gegen blinde Masquerade-Attacken. Es ist praktisch

unmöglich für einen blinden Angreifer (der keinen Zugriff auf die Transportverbindung hat) eine Verbindung vorzutäuschen (connection forgery), da er dafür den richtigen Validierungstag erraten müsste. Eine zusätzliche Funktion dieses Datenfelds stellt die Erkennung von Paketen aus einer alten Verbindung zwischen den gleichen Teilnehmern dar.

Man darf den Validierungstag nicht als absoluten Schutz gegen Angreifer betrachten, denn es funktioniert nur gegen absolut blinde Attacker (s. Abbildung 3). Falls der Angreifer die Pakete lesen kann (s. Abbildung 3b), kann er den Validierungstag einfach kopieren. Für eine bessere Kommunikationssicherung ist der Nutzer auf IPsec, SSL/TLS angewiesen [11].

4. VERGLEICH MIT TCP UND ANWENDBARKEIT

Nachdem wir die Fähigkeiten von SCTP schematisch beschrieben haben, versuchen wir in diesem Abschnitt eine Gegenüberstellung zu TCP herzustellen und dabei die Stärken und Schwächen von SCTP zu erkennen.

4.1 Chunks vs. Bytestream

Wie in Abschnitt 2.3 erwähnt, ist SCTP Message-orientiert im Gegensatz zu TCP, das Byteströme vermittelt. Das hat sowohl Vorteile als auch Nachteile. In SCTP muss sich der Nutzer nicht um die Segmentierung der gesendeten Daten kümmern. Da das ULP (Upper-Layer-Protocol) jetzt keinen Bytestrom mehr abfragt, sondern ganze Nachrichten, entsteht eine neue Möglichkeit der Kommunikation zwischen Transport- und Anwendungsschicht. Die überliegende Instanz wird benachrichtigt sobald eine neue Nachricht komplett empfangen wurde. Das macht die explizite Nachfrage an Daten überflüssig seitens der Anwendung, denn der Nutzer kann einfach im Idle Modus warten, bis er auf angekommene Daten benachrichtigt wird. Bei TCP werden Anwendungen ebenfalls blockiert, falls keine Daten im Puffer liegen, aber es ist schwierig eine Anwendung zu implementieren, die aus mehreren TCP-Verbindungen gleichzeitig lesen kann (vgl. SCTP-Streams in Kapitel 3.3).

Andererseits entsteht ein zusätzlicher Overhead bei dieser Art der Datenvermittlung, da jede Nachricht einen eigenen Header (4 Byte) hat.

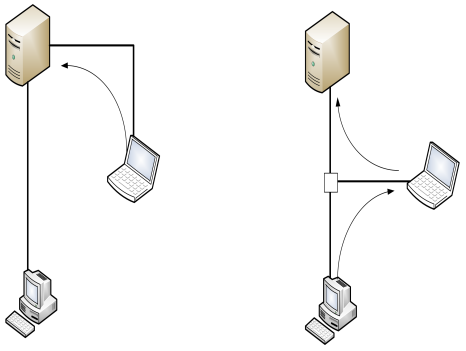


Abbildung 3: Links – ein absolut blinder Angreifer, rechts – der Anreifer kann die Kommunikation mitschneiden, aber nicht beeinflussen

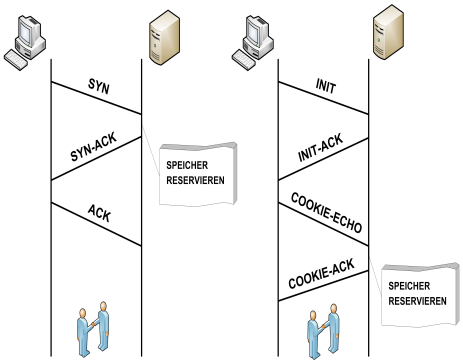


Abbildung 4: Verbindungsaufbau bei (a) TCP und (b) SCTP

4.2 Vor- und Nachteile der Streams

Streams (s. Abschnitt 3.4.3) sind gut geeignet für den Transport von mehreren Date(ie)n, deren Bytereihenfolge wichtig ist, nicht aber die Reihenfolge ihrer Ankunft. Es ist zum Beispiel unerheblich, welches Bild einer Web-Seite zuerst ankommt, es ist aber nicht egal, in welcher Reihenfolge die einzelnen Bytes eines Bildes ankommen. Eine Web-Seite ist ein gutes Beispiel, wie Streams effektiv eingesetzt werden können. Die ganze Web-Seite kann innerhalb einer Assoziation übertragen werden, wobei die (x)HTML Datei, die Bilder, Skripte, Style-Sheets, Applets etc. jeweils in eigenen Streams übertragen werden können. Wenn sich dann ein Paket auf der Strecke verzögert oder gar ausfällt, wird das die restlichen Komponenten nicht blockieren. [6] bietet eine gute praktische Evaluation der Streams in SCTP und untersucht statistisch, inwiefern sie die Performanz einer Verbindung beeinflussen.

4.3 Das Cookie

TCP stellt eine Verbindung mittels 3-Way-Handshake her: Der Client sendet ein SYN-Paket, der Server antwortet mit SYN-ACK und der Client sendet ACK zur Bestätigung der Verbindung (s. Abbildung 4a). Das hat den Nachteil, dass zu jedem SYN-Request auf der Serverseite Speicher für die entstehende Verbindung zugewiesen wird. Der in diesem Speicher enthaltene Transport Control Block (TCB) wird für eine Zeitlang behalten, in der aber ein potentieller Angreifer sehr viele verfälschte TCP-Verbindungen anfordern kann. Manche Server implementieren das sogenannte Cookie-Verfahren, um sich gegen solche Angriffe zu verteidigen: Wenn sich der Puffer auf dem Server füllt, teilt er keinen Speicher mehr zu, sondern kodiert die Verbindungsparameter in einem sog. SYN-Cookie im SYN-ACK-Paket und schickt es an den Anfrager zurück. Diese Technik verhindert Denial-of-Service-Angriffe gegen den Server, schränkt aber die Verbindung ein (die maximale Segment-Größe ist z.B. begrenzt), da das Cookie nur 32 Bit lang sein darf. Ein weiterer Nachteil ist, dass die Verbindung einfriert, wenn das ACK des Clients verloren geht, da der Server nicht weiß, dass eine Verbindung hergestellt wurde und deshalb die eingehenden Daten des Clients schweigend verwirft (s. Abbildung 5). Referenz [2] beschreibt das Verhalten von TCP unter SYN-Angriffen sowie den Cookie-Mechanismus.

In SCTP ist dieses Verfahren ein Teil des Verbindungsauf-

baus (s. Abbildung 4b). Neben dem Schutz vor DoS-Angriffen entlastet es den Server und generiert kaum zusätzlichen Overhead. Das Cookie ist mit einer Checksumme (HMAC) geschützt, die mit dem Private Key des Servers berechnet wird, kann also vom Empfänger/Angreifer praktisch nicht erraten bzw. verändert werden.

4.4 Prüfsumme

Die von SCTP verwendete Prüfsumme wurde in seiner ersten Version mit dem Adler-32-Algorithmus [13] berechnet. [14] definiert diese Vorschrift neu und spezifiziert den crc32-Algorithmus als Standard, weil das für kleinere Pakete erheblich besseren Rauschschutz bietet.

4.5 Ähnlichkeiten von SCTP und TCP

Die zwei Transportprotokolle sind sich in vielen Punkten ähnlich oder gleich. Zum Beispiel verwenden beide den gleichen SACK-Ansatz, um Daten rückzumelden (SACK ist bei TCP jedoch lediglich als Extension realisiert [4]). Das Portsystem ist das Gleiche und die Stau- und Flusskontrolle basiert auf den gleichen Prinzipien, was eine Kooperation ermöglicht, falls beide Protokollinstanzen parallel laufen: z.B. ist es möglich, dass beide Protokolle die gleiche Instanz der Stau- und Flusskontrolle verwenden.

5. PROBLEME UND EINSCHRÄNKUNGEN VON SCTP

Leider hat SCTP nicht nur Vorteile, sondern auch Probleme. Hier werden einige der SCTP-Schwächen behandelt, die auf die behandelten Features zurückzuführen sind.

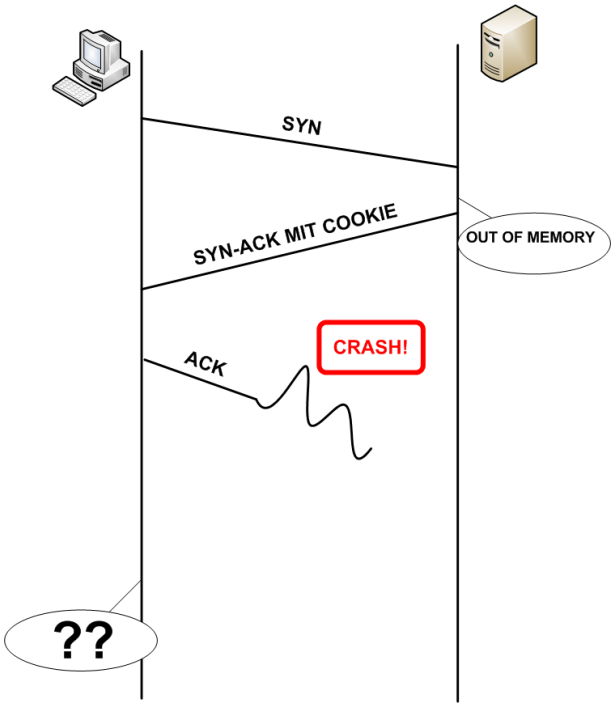


Abbildung 5: Eingefrorene Verbindung mit verlorenem TCP Cookie

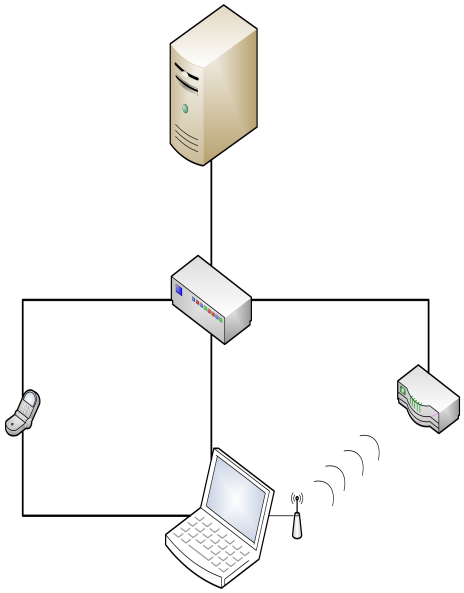


Abbildung 6: Obwohl der Client mehrere Netzwerkschnittstellen hat, verlaufen alle über den gleichen Router

5.1 Performanz

Eine SCTP-Assoziation mit nur einem Stream transportiert Daten langsamer, als eine gewöhnliche TCP-Verbindung. Die Einteilung des Datenstroms in Messages generiert zusätzlichen Overhead.

5.2 Multihoming

Leider kommt es vor, dass Endpunkte A und B mehrere Netzwerkschnittstellen besitzen können, die Pfade aber verlaufen über ein und denselben Knoten innerhalb des Netzwerks und können bei einem eventuellen Ausfall dieses Knotens alle betroffen sein (s. Abbildung 6). Multihoming ist mehr eine Verbesserung der Ausfallsicherheit, als Garantie gegen den Ausfall der Transportwege.

Ein weiteres Problem ist IPsec (VPN) in Zusammenarbeit mit SCTP. Falls die Endpunkte jeweils m und n IP-Adressen haben, muss IPsec insgesamt $2 * m * n$ Sicherheitsassoziationen aufbauen. In [21] wird diskutiert, wie IPsec angepasst werden kann, um eine Multihoming-Verbindung zu erkennen.

5.3 Inkompatibilität mit NAT und Firewalls

Da ein SCTP-Endpunkt mehrere IP-Adressen haben kann, ist es genau von jenen Internet-Verfahren betroffen, die IP-Adressen verbergen, blockieren oder verändern, z.B. NAT [22]. An dieser Stelle soll nicht zu tief ins Thema eingetaucht werden, da Network Address Translation und Firewalls komplexe Instanzen sind, die sich nicht nur auf den IP-Level beschränken. Die einfachste Methode, SCTP über NAT durchzusetzen ist, indem man beim Verbindungsaufbau keine explizite IP-Adressen an den Peer mitteilt. Das bewirkt, dass die Adresse im IP-Header bezogen wird; die IP-Pakete werden vom NAT erkannt und richtig umgeleitet. Intelligente NAT-Geräte könnten SCTP-Pakete erkennen und die IP-Adressen richtig übersetzen. [15] beschreibt

die Einschränkungen, die NAT auf SCTP erzwingt, und gibt eine NAT-Implementierung an, die SCTP unterstützt.

Bei Firewalls liegt das Problem darin, dass SCTP noch nicht weit verbreitet ist, es ist also unwahrscheinlich, dass eine Firewall SCTP-Pakete durchlässt.

6. VERWANDTE ARBEITEN

Es gibt viele Artikel im World Wide Web, die SCTP mit anderen Transportprotokollen vergleichen und den Nutzen bewerten.

Die Masterarbeit von Ivan Rodriguez [8] bietet einen breiten Hintergrund zum Thema SIGTRAN, SS7 und SCTP und behandelt die hier eingeführten Konzepte detailliert (und übrigens auch in einer unterhaltsamen Form).

[3] befasst sich mit der Anwendbarkeit von SCTP in MPI (ein Netzwerkinterface für wissenschaftliche und andere hochparallele Rechenanlagen).

[7] bietet eine Möglichkeit, SCTP Multihoming zu verwenden, um Daten gleichzeitig über alle verfügbaren Pfade zu transportieren (Load Sharing).

7. ZUSAMMENFASSUNG

7.1 Ausblick – welches Potenzial hat SCTP, sich als Standard zu etablieren?

SCTP ist im Jahr 2000 als das vierte Transportprokoll neben TCP, RTP und UDP von der IETF standardisiert worden und es wird bereits von mehreren Betriebssystemen unterstützt. Es gibt eine Implementierung für den Linux-Kernel, die man auf Sourceforge herunterladen kann [16]. Die Implementierung von Randy Stewart ist im KAME-Project [17] integriert und IBM arbeitet an einer Version für AIX.

IPsec arbeitet derzeit daran, Overhead im multihomed Transport zu reduzieren [18]. Am Thema „TLS over SCTP“ wird auch gearbeitet [11]. Es ist möglich, SCTP hinter der TCP API zu verbergen, um die Einführung des neuen Transportprotokolls zu beschleunigen [19]. Es gibt auch eine Implementierung, die SCTP über UDP tunnelt.

Obwohl SCTP auf TCP basiert, das ein stabiles Verhalten über längere Zeit aufgewiesen hat, ist SCTP noch nicht weit verbreitet und dadurch auch nicht Subjekt von Angriffen und Anomalien. Es könnte durchaus andere, neue Schwächen und Probleme aufweisen, die TCP und UDP nicht haben.

7.2 Schlussfolgerung

SCTP hat die Aufgabe, einen besseren, flexibleren, zuverlässigeren Transportdienst für die Bedürfnisse des Internets in der nahen Zukunft anzubieten. Es soll die Einfachheit und Stabilität von TCP mit der Performanz von UDP kombinieren und diese mit etwas Ausfallsicherheit zu erweitern. Mit der Geschwindigkeit, mit der sich das Internet entwickelt, wird es nicht mehr lange dauern, bis Begriffe wie Multihoming, Parallel Data Transmission und Network Resilience eine globale und wichtige Rolle spielen werden. In diesem Paper wurde die Umsetzung dieser Konzepte in SCTP von einem theoretischen Standpunkt aus betrachtet; ob sie aber effizient, stabil und sicher implementiert sind, kann nur die Praxis zeigen.

8. LITERATUR

[1] Stewart, R., *„Stream Control Transmission Protocol“*, RFC 4960, IETF, September 2007, `http://tools.ietf.org/html/rfc4960`

[2] Nakashima, T., Sueyoshi, T., *„Performance Estimation of TCP under SYN Flood Attacks“*, Complex, Intelligent and Software Intensive Systems, 2007. CISIS 2007. First International Conference, pp.92-99, 10-12 April 2007 doi: 10.1109/CISIS.2007.48

[3] Kamal, H., Penoff, B., Wagner, A., *„SCTP versus TCP for MPI“*, Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference, pp. 30- 30, 12-18 November 2005 doi: 10.1109/SC.2005.63

[4] Mathis, M., Mahdavi, J., Floyd S., Romanow, A., *„TCP Selective Acknowledgment Options“*, IETF, October 1996, `http://tools.ietf.org/html/rfc2018`

[5] Mogul, J., Deering, S., *„Path MTU Discovery“*, RFC 1191, IETF, November 1990, `http://tools.ietf.org/rfc/rfc1191`

[6] Grinnemo, K.-J., Andersson, T., Brunstrom, A., *„Performance Benefits of Avoiding Head-of-Line Blocking in SCTP“*, Autonomic and Autonomous Systems and International Conference on Networking and Services, 2005. ICAS-ICNS 2005. Joint International Conference, pp.44-44, 23-28 October 2005 doi: 10.1109/ICAS-ICNS.2005.73

[7] Abd El Al, A., Saadawi, T., Myung Lee, *„Bandwidth aggregation in stream control transmission protocol“*, Computers and Communications, 2004. Proceedings. ISCC 2004. Ninth International Symposium, vol.2, pp. 975-980, 28 June-1 July 2004

[8] Rodríguez, I. A., Kantola, R., Loughney, J., *„Stream Control Transmission Protocol. The design of a new reliable transport protocol for IP networks“*, Helsinki University of Technology, February 2002

[9] Allman, M., Paxton, V, Stevens, W., *„TCP congestion control“*, RFC 2581, IETF, April 1999, `http://www.ietf.org/rfc/rfc2581`

[10] Jiemin Liu, Hongxing Zou, Jingxin Dou, Yuan Gao, *„Reducing Receive Buffer Blocking In Concurrent Multipath Transfer“*, Circuits and Systems for Communications, 2008. ICCSC 2008. 4th IEEE International Conference, pp.367-371, 26-28 May 2008 doi: 10.1109 / ICCSC.2008.85

[11] Jungmaier, A., Rescorla, E., Tuexen, M., *„Transport Layer Security over Stream Control Transmission Protocol“*, RFC 3436, IETF, September 2007 `http://www.ietf.org/rfc/rfc3436`

[12] Jungmaier, A., Pathgeb, E., *„On SCTP multi-homing performance“*, Telecommunication Systems, Springer Netherlands, vol. 31, no. 2-3 / March 2006

[13] Deutsch, P., Gailly, J-L., *„ZLIB Compressed Data Format Specification version 3.3“*, RFC 1950, IETF, May 1996 `http://tools.ietf.org/html/rfc1950`

[14] Stone, J., Stewart, R., Otis, D., *„Stream Control*

Transmission Protocol (SCTP) Checksum Change“, RFC 3309, IETF, September 2002 `http://tools.ietf.org/html/rfc3309`

[15] Tuxen, M., Rungeler, I., Stewart, R., Rathgeb, E., *„Network Address Translation for the Stream Control Transmission Protocol“*, Network, IEEE , vol.22, no.5, pp.26-32, September-October 2008 doi: 10.1109 / MNET.2008.4626229

[16] *„Linux Kernel SCTP“*, `http://sourceforge.net/projects/lksctp/Df`

[17] *„The KAME Project“*, `http://www.kame.net/`

[18] Cano, M-D., Romero, J. A., Cerdan, F., *„Experimental Tests on SCTP over IPSec“*, Network and Parallel Computing, 2008. NPC 2008. IFIP International Conference, pp.96-102, 18-21 October 2008 doi: 10.1109 / NPC.2008.92

[19] Bickhart, R., Amer, P., Stewart, R., *„Transparent TCP-to-SCTP Translation Shim Layer“*, EuroBSDCon 2007, Copenhagen, 5/07 `http://www.cis.udel.edu/~amer/PEL/poc/pdf/EuroBSDCon2007-bickhart-SCTP-Shim-layer.pd`

[20] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., Conrad, P., *„Stream Control Transmission Protocol Partial Reliability Extension“*, RFC 3758, IETF, May 2004 `http://www.ietf.org/rfc/rfc3758`

[21] Bellovin, S., Ioannidis, J., Keromytis, A., Stewart, R., *„On the use of SCTP with IPsec“*, RFC 3554, IETF , July 2003 `http://www.ietf.org/rfc/rfc3554`

[22] Egevang K., Francis P., *„The IP Network Address Translator (NAT)“*, RFC 1631, IETF, May 1994, `http://tools.ietf.org/html/rfc1631`

Google Big Table

Xiao Chen
Betreuer: Marc-Oliver Pahl
Seminar Future Internet SS2010
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: Cx3606@gmx.de

ABSTRACT
Bigtable is a storage system for structured or semi-structured data [1]. Bigtable can be regarded as a distributed, non-relational Database from a system point of view (Bigtable is using a different data model than relational Databases). Bigtable scales to large data sets and provides a good performance of accessing the data. As of January 2008, there are more than sixty Google applications using the Bigtable as storage provider, such as Google Earth, Web-Indexing and Personalized Search. Bigtable fits different demands of those applications. It addresses problems which cannot be handled by standard relational databases. In this paper, we give a fundamental overview of Bigtable's design and implementation. We will describe the differences between Bigtable and relational database and focus on the different data models used by them.

Keywords
Bigtable, non-relational database, distribution, performance.

1. INTRODUCTION
The development of the Internet has introduced many new Internet based applications. The web-index and Google earth are for example used by millions of users from Internet. To manage these terabytes data (Google Earth has more than 70 terabyte data) becomes a challenge. To address the demands of those applications, Google has started the development of Bigtable in 2003. The Bigtable is designed to store structured or semi-structured data in nodes that are distributed over network.

The project is steadily growing since then. As of January 2008, there are over 600 Bigtable clusters at Google [2] and over sixty productive applications based on it. The design goal of the Bigtable mainly focuses on four aspects: high scalability, high availability, high performance, and wide applicability. There are some database models like "Parallel databases" [3] providing speed up and scale up of relational database queries, but Bigtable distinguishes itself from those models: it is not a relational database. It provides a different interface as those relational database models.

Bigtable shares many database strategies: Data scan, Data Storage and Data access. Unlike a relational database which stores a fixed schema in database server, the data logic is embedded in the client code: the client code controls dynamically how the data structure is.

Bigtable relies very much on a so called mapreduce job to guarantee the design goal of a high performance of the distribution. Mapreduce [4] is a framework for processing and generating large data sets. Bigtable is used as input or output source for Mapreduce jobs. Mapreduce Job provides a very fast transformation for the data of Bigtable to hundreds of nodes across the network.

In Section two, we firstly introduce an application example which is using Bigtable as storage provider. We are going to see the potential requirements of this application and why the standard Relational Database Management System (RDBMS) cannot be used for this application. We will explain briefly how Bigtable solves those demands from a high level design strategic perspective. Section three contains information of differences between the RDBMS and the Google Bigtable. We will introduce the data model used by the two Database models. This will give a further explanation why Bigtable is more suitable to store large datasets in a distributed way. Section four will provide an overview of the building blocks of Bigtable. Section five introduces the basic implementations. In section six we describe some refinements Bigtable is using to archive the design goals. In Section seven we will give a short overview of the client API. Section eight presents the entire architecture of Bigtable and our conclusions of the design principles.

2. APPLICATION EXAMPLE: GOOGLE EARTH
Google Earth is one of the productive applications which are using Bigtable as storage provider. It offers maps and satellite images of varying resolution of the Earth's surface. Users can navigate through the earth surface, calculate a route distance, execute complex or pinpointed regional searches or draw their own routes. In following section, we introduce some fundamentals of the implementations. We will see why Bigtable can address the requirements of Google Earth better than a standard relational Database.

Google Earth is using one table to preprocess raw data, and several other tables for serving the client data. During preprocessing, the raw imagery is cleaned and consolidated into serving data (the final data used by the application). The preprocessing table stores the raw imagery. It contains up to 70 terabytes data and therefore cannot be maintained in the main memory. It is served from the disk. The imagery was already be consolidated efficiently, for this reason Bigtable compression is disabled. The details for Bigtable's compression methods can be found on section 6.1 compression.

The size of the preprocessing table is the first reason why we cannot use RDBMS to store the data, the 70 terabytes data cannot be stored as one table hosts in a single machine.

The serving system of Google Earth is using a single table to store the index data. Although the index table is relative small (500GB), if we use one machine to host the table we still have to move the data to hard disk. For performance considerations, we cannot implement it, because the index table must serve tens of thousands of queries per second per datacenter with low latency. With the data stored on disk, we would have no chance to fulfill the requirements with the current hardware technology. We need a solution to distribute the data into multiple nodes.

A major characteristic of Bigtable is its scalability. Relational Database also scales but only in single node. When the hardware capacity of the single node is reached, the load needs to be distributed to other nodes. Some Databases like Oracle provide services like replication jobs to address scale loads out of a single machine. For an application like Google Earth which has a massive workload, it will require hundreds or thousands of nodes' capacity to store the data. Standard replication jobs cannot be used in this kind of situation. RDBMS is more suitable for applications which are hosted on one single node but Bigtable is designed for data distribution of a large scale into hundreds or thousands of machines over network. Mapreduce Job [4] is always used for Bigtable applications to distribute and to process the data to nodes and within network.

By speaking of the scalability requirements, another consideration is the flexibility. This can also become a problem of managing the system if we only have RDBMS located on a single node. Taking the index table used by Google Earth as example, when the server load or the size of the index table becomes double and the hardware capacity of this single node is reached we cannot upgrade the hardware on the single node as fast as the speed of the change. Bigtable allows managing the data in a flexible way to add or remove a node from the distribution cluster. More details about how Bigtable manages the tablets assignment can be found in section 5.3.

3. DIFFERENT DATA MODEL USED BY BIGTABLE AND RDBMS

The example of Google Earth presents the motivations of using Bigtable. We will explain the different Data Models used by Bigtable and RDBMS. This will on one hand demonstrate why Bigtable is not a relational database system. On the other hand, it will give explanations why Bigtable is more suitable for applications which require distributed storage.

A relational database is a collection of tables (entities). Each table contains a set of columns and rows. The tables may have constrains to each other and relationships between them.

Figure 1 shows a typical Data model used by RDBMS. The column Id from entity "Functionary" is used as foreign key which is referenced to column "idSupport" of entity "Support". The column "idAttendee" of entity "Attendee" is referenced to column "AttendeeId" of Entity "Support". The relationship (Data logical) between "Functionary" and "Attendee" is thus kept in entity "Support".

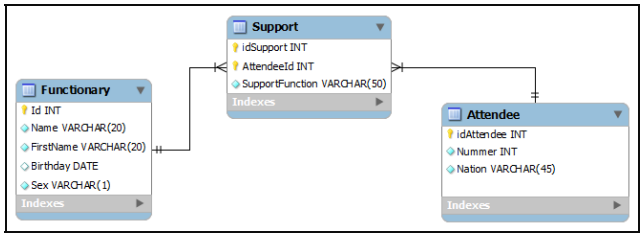


Figure 1. relational Data Model: functionary has constrains with Attendee via Support entity.

The RDBMS model exists since almost 30 years. When it was developed, the RDBMS was not widely used due to hardware limitations. Even a simple select statement may contain hundreds of potential executing paths which the query optimizer needs to calculate at runtime. Today, the hardware can satisfy the demands of RDBMS, so the relational Database has become a dominant choice for common applications. They are easier to understand and to be used - although it still provides less efficiency compared to the legacy hierarchically database. Almost all databases we are using now are RDBMS; typical examples are Oracle, MS Sql and DB2.

Comparing to RDBM, which stores the data logical within table itself, Bigtable is a sparse, distributed, persistent multi-dimensional sorted map. The map is indexed by a row key, column key, and a timestamp. It can thus be seen as a 3D table in form of row*column*timestamp. Each value in the map is an un-interpreted array of bytes. The data model used by Bigtable can be concluded as follow formula:

(row: String, Column: String, time: Int64) → String

The row keys are a user-defined string. The data is maintained in Bigtable in a lexicographic order by the row key.

Column key syntax is written in the form of "Family:optional_qualifier". The Column keys are grouped together into the Column families. Column families can be regarded as categories of its column. The data belongs to the same Column family is compressed together with the column information. The Column families are the basic unit of accessing the data. They can also have attributes/rules that apply to their cells, such as "keep n time entries" or "keep entries less than n days old".

The timestamp is used to version the data, so we can track the changes of the data over the time. The size of the timestamp is a 64bit integer. Each column family can have different number of versions. For example, a data string in Bigtable has two column families: content and anchor. Content column family contains the page of content, and the anchor column family contains text of anchors which references to the page. The two column families can have different number of versions based on application's requirement. The content column family can keep the latest 3 updates while the anchor column family only keeps the latest update.

The timestamp can be used together with the rules added to Column to manage the lifecycle of the date. The old timestamps could be removed by a garbage-collection.

After this short introduction to the datamodel used by Bigtable, we can now see that its data model does not equal the Data Model used by a relational database. Bigtable characterizes itself as a database management system which is commonly called a key/value Database [6], the name is not official, and some documentation also refer to this kind of database as distributed database or distributed hashtable.

4. BUILDING BLOCKS

Bigtable cluster contains one or several tables. Each table consists of set of tablets by the range of the row key. The tablets are usually around 100-200MB and can be distributed into different nodes across the network. Each node(tablet servers) saves about 10~1000 tablets wtihin its Google File System(GFS) [7]. Figure 2 shows a simpfied overview of the structure of the Bigtable implementation and the major building blocks are used there: Google File System (GFS), SSTable (sorted string table) and Chubby.

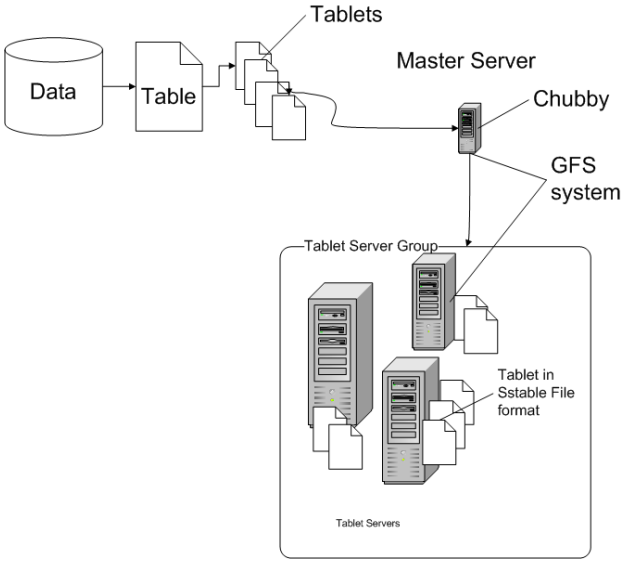


Figure 2 Building Blocks: Chubby, GFS, SStable

GFS is a File system used by Bigtable to store the logs and files. A Bigtable cluster usually runs a shared pool of the distributed nodes. Bigtable also requires a cluster management system (CMS). The CMS is used to schedule jobs, manage resource on shared machines, replicate jobs of failure machines and monitor the machine status.

SSTable stands for "sorted string Table", this is a immutable file format which Google internally used to save the Tablets. The paper [5] of the Bigtable provides follow information for SSTable:

"An SSTable provides a persistent, ordered immutable map from keys to values, where both keys and values are arbitrary byte strings. Operations are provided to look up the value associated with a specified and to iterate over all key/value pairs in a specified key range."

Internally, the SSTable contains a set of blocks which is 64KB. This size can be configurable by application requirements. On the end of the SSTable an index will be generated and it will be loaded into memory when an SSTable is opened. In this way, a lookup can be done using single disk seek: we just need to find block using the index stored in memory and then read the block from the disk. Figure 3 summarize the implementations of the SSTable:

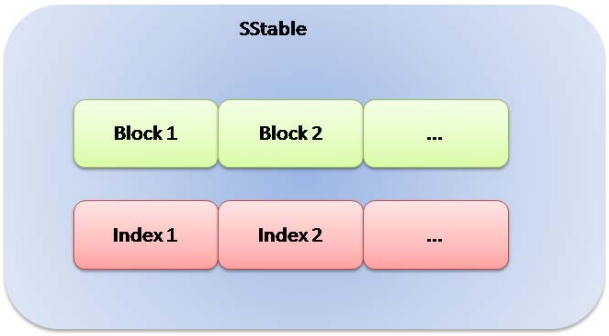


Figure 3 SSTable schema: index of each block will be loaded in memory. So the lookup to the actual data only needs one disk seek.

Alternatively, an SSTable can be completely copied to the main memory to avoid read from disk.

Bigtable depends on a highly-available and persistent distributed lock service called Chubby [8]. It is used to keep track of tablet servers in Bigtable. Chubby itself is a cluster service that maintains five active replicas, one of which is the master. The service is running when majority of the replicas are running and can communicate with each other. As explained in the document of Bigtable [5], Chubby provides a namespace that consists of directories and small files. Each directory or file can be used as a lock, and reads and writes to a file are atomic. The Chubby files are cached consistently in the Chubby client library. Chubby allows the client to take a local lock, optionally with the metadata which is associated with it. When a client session has been expired, Chubby will revoke all locks that it has. The client can renew the lock by sending "keep alive" messages to Chubby.

Chubby can be seen as the global lock repository of Bigtable. How Bigtable is using the Chubby can be seen by the section 5.3 tablet assignment.

5. IMPLEMENTATION

5.1 Major Components

The Bigtable implementation contains three major components: A Client library, one Master Server and many Tablet Servers.

The master server is responsible for assigning tablets to the tablet server. It detects the additional and expiration of the tablet server in order to balance the tablet server load. It also does the garbage collection of the files on the GFS. Furthermore, when the schema of the rows and column families need to be changed, the Master Server also manages those changes.

Tablet server is designed to manage the set of tablets. It handles the read and write requests to the tablet. When a tablet is growing too large, the Tablet server also splits it into small tablets for the

future processing. Tablet servers can be added or removed from a Bigtable cluster based on the current workload. This process is managed by master server.

Although there is a one single master server existing in the cluster, since the clients do not rely on master server for the tablet location information, the load of the master server is very low. The client communicates with the tablet server directly to read or write data from a tablet.

5.2 Tablet Location

Bigtable is using a three-level hierarchy to store the tablet location analogous to that of a B+ [9] tree: Root tables, metatables and usertables (Figure 4)

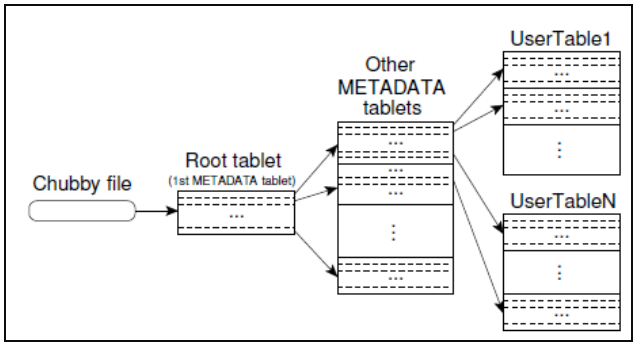


Figure 4 Table location hierarchies based on Bigtable paper [5]

Chubby stores a file which contains location information to a root tablet. The root tablet contains location information to other metadata tablets in a special Metadata tablet. This Special Metadata tablet will never be split – to ensure the location hierarchy is never expanded more than three levels.

Each client library contains a cache to the location information. If the cache is empty or if the client detects that the cache information is not correct, the client will move up the hierarchy to retrieve the location recursively. In worst case, a location look up through the three-level hierarchy may require six network round trips including a lookup in the Chubby file.

Bigtable stores secondary information in the metadata table like logs (when a server begins serving it), such information is helpful for troubleshooting or performance analytics.

5.3 Tablet Assignment

A tablet server contains a set of tablets. Each tablet can be assigned to one tablet server at one time. The master server is used to keep tracking the set of live tablet servers, managing the current assignments of tablets to tablet servers. When a tablet becomes unassigned, the master server will verify firstly if there is enough space on the live tablet server, it will then assign the unassigned tablet to this tablet server.

The Chubby service which we mentioned in section 4 is used by master server to keep track of tablet servers.

When a tablet server starts up, it creates, and gets an exclusive lock on a unique-named file in the Chubby directory [5]. This directory is monitored by the master server, so the master can discover the newly arrived tablet servers.

When the tablet server loses the exclusive lock on the directory, it stops serving. The tablet server will reacquire the exclusive lock of the file as long as it still exists in the directory. If the file does not exist, the tablet server will kill itself.

When the tablet server terminates itself, the tablets which are associated within the tablet server will be unassigned, because the tablet server will attempt to release its lock. The Master can then reassign those unassigned tablets as soon as possible.

The master asks the lock status of the tablet server periodically. If the tablet server replies with a loss of the lock status, or the master does not get a reply from a certain tablet server after several attempts, the master server will try to acquire an exclusive lock on the file itself. If the file can be locked, it implies that Chubby is alive and the tablet server which locked this file died. The master will ensure that this tablet server can't serve any data again by deleting the server file.

Once the server file is deleted, those tablets which are previously assigned to this tablet server become unassigned.

As mentioned in section 4, Google uses the CMS system to manage the clustered nodes. When a master server is started by the CMS, it needs to be informed of current tablet assignment. In the following step the assignment information is collected:

1. Initialize a unique master lock in Chubby to prevent other concurrent master server from initializing.
2. Scan the server file directory to recognize the live tablet servers.
3. Tell the master server to connect to each live tablet server to scan the tablets which are assigned to the tablet server.
4. The master scans the metadata table to learn the set of unassigned tablets.

5.4 Implementation Memtable

Memtable is the in-memory representation of the most recent updates of a tablet. It contains the recently committed logs which are stored in memory in a sorted buffer.

The memtable will not increase infinitely. When it reaches its threshold (depends on the main memory size), the current memtable will be converted to an SSTable and moved into the GFS, a new memtable will also be created. This process is called Compaction. Those SSTables act as snapshots of this server so they can be used for recovery after a failure.

When a write option arrives, the tablet server checks the well-formedness and the authorization to see if an option is permitted and stores it into the commit log. After the write has been committed, the content is inserted into the memtable.

When a read option arrives, the tablet server will also check if the option is well formed and if the sender of the option is authorized. If it is a valid operation, the operation is executed in a merged view of a sequence of the SSTable and the Memtable. Since the SSTable is sorted in a lexicographically way it is easy to form the merged view.

When those tablets are splitting or getting merged, it does not lock the read/write operations.

Figure 5 is made to represent how read and write is done and procedure of the Memtable.

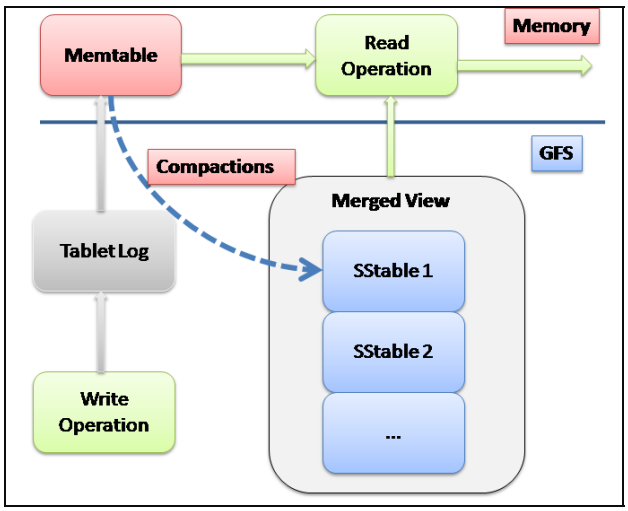


Figure 5 read write process, the read process is performed on a merged view of the SSTable. The write operation is written into the Tablet Log. When Memtable increases its size, it will be converted to SSTables and moved to GFS

6. REFINEMENT USED BY BIGTABLE

6.1 Locality groups

Locality Groups are user to group parts of data together which have similar user criteria. For example, the metadata of a webpage can be grouped together as one locality group, while the content of the webpage is grouped as another locality group

6.2 Compression

The Bigtable implementation relies on a heavy use of the compression. Clients can specify whether or not SSTables for a locality group are compressed or not. Client also specifies which compression schema is to be used.

A typical two-pass compression schema is used by many clients. The first pass uses Bentley and McIlroy's scheme [10], which is designed for compressing very long strings. The second pass looks into the small 16kb window for repetitions of the data. The first and second pass is done in a very quick way, the encoding cost is 100-200MB/s and the decoding cost 400-1000MB/s. Even those two schemata are chosen for a quick decoding and encoding process. In t practice they provide a high rate as 10 to 1 of the compression.

6.3 Merging unbounded SSTables

One optimization used in Bigtable is merging of unbounded SSTables. A single SSTable is merged from SSTables for a given tablet periodically. This single SSTable contains also a new set of updates and index. This will prevent that the read option loading every data from this small piece of SSTable and access the GFS many times.

6.4 Caching

The caching is mainly used to improve the read performance. There are two levels of caching used by Bigtable: scan cache and block cache. Scan cache is a high level cache which caches the key-value pairs returned by SSTables. It is most useful for applications which are reading data repeatedly. The Block Cache is a low level cache. It is useful for applications which read Data close to data they recently read.

6.5 Bloom filter

A very important problem by using Bigtable is the access of the SSTable files. As mentioned in section 5, the SSTable is not always kept in memory, the user read operation may need many accesses in the GFS (located in the hardware layer) to load the state of the SSTable files. The paper of Bigtable [5] explains that they reduce the number of accesses by allowing clients to specify that Bloom filters [11] should be created for SSTables in a particular locality group. Bloom filter is an algorithm which is used to verify if a data is in the membership of the set. In the implementations of Bigtable, the bloom filter is kept in memory in order to probabilistic if a data exists in a given row/column pair. The memory usage to store the bloom filter is very small, but this technique drastically reduces the access of the data in disk.

7. API

In the relational database, the data will be updated, inserted, deleted using SQL Statements. Bigtable does not support SQL (it supports a Google designed language called Sawzall [12] to filter the data). Client applications can write or delete data, lookup values from rows and column families within Bigtable using API Method calls. The application and data integrity logic is thus contained in the application code (not like the relational data, the embedded logic is stored in the Data model with triggers, stored procedure, etc.).

As mentioned in section 2, Bigtable is storing the data in form of row*column*timestamp, the column family is the category of the column. Figure 6 shows a simple API code written in C++ to modify the data stores in Bigtable.

```
// Open the table
Table *T = OpenOrDie("/bigtable/web/webtable");

// Write a new anchor and delete an old anchor
RowMutation r1(T, "com.cnn.www");
r1.Set("anchor:www.c-span.org", "CNN");
r1.Delete("anchor:www.abc.com");
Operation op;
Apply(&op, &r1);
```

Figure 6 Open a table and write a new anchor, which is a column family, and then delete the old anchor. [5]

8. CONCLUSION

Taking account of the above, Figure 7 shows a simplified overview of the Bigtable's Architecture.

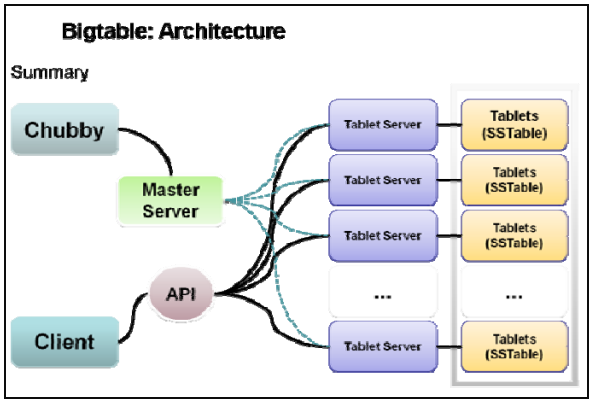


Figure 7 An overview of the Bigtable Architecture

Bigtable varies from traditional relational database on its data model. Bigtable tends to be used by applications like Google Earth, which require a large storage volume. Legacy network database or relational database can not address the requirements by those kinds of applications to distribute data over thousands of hosts. A relational database system is more powerful and is still a dominant choice for those applications which require a storage that is hosted by several hosts.

On the other hand, large distributed systems are more vulnerable to many types of failures: memory and network corruptions, large clock skew and Chubby service failures. All those problems could cause Bigtable implementations to fail. Some of those problems are addressed by changing various protocols used by Bigtable, but the implementations still need further refinements.

Here are some design principles which can be extracted from the Bigtable implementation:

- Use single master server for a quick, simple management of distribution.
- Use refinements technique to avoid accessing the disk directly. The latencies caused by read operations will be more expensive as the network round trips.
- Replicate the storage to handle the failure of cluster node.
- Avoid replicating the functionalities: it is more expensive to keep the server in sync as to replace a failed server. So we should not replicate the functionalities in different server.
- Make a high available rate of the communication/lock service. Chubby is an example: If this service fails, most Bigtable operations will stop working.

9. REFERENCES

[1] **Buneman, Peter.** *" Tutorial on semi-structured data. "*, 1997.

[2] **Wilson Hsieh, Jayant Madhavan,Rob Pike.** *" Data management projects at Google. "*, Chicago, IL, USA : ISBN:1-59593-434-0 , 2006 . ACM SIGMOD international conference on Management of data. S. 36.

[3] **David Dewitt, Jim Gary.** *" Parallel database systems: the future of high performance database systems. "*, 6, New York, NY, USA : ACM, 1992, Bd. 35. ISSN:0001-0782 .

[4] **Jeffrey Dean, Sanjay Ghemawat.** *" MapReduce: Simplified Data Processing on Large Clusters. "*, 2004. OSDI '04 Technical Program. S. 1.

[5] **Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber.** *" Bigtable: A Distributed Storage System for Structured Data. "*, WA : s.n., 2006. OSDI'06: Seventh Symposium on Operating System Design and Implementation. S. 1.

[6] **Bain, Tony. "** http://www.readwriteweb.com/enterprise/2009/02/is-the-relational-database-doomed.php. **",** *" ReadWrite. "*, [Online] 12. February 2009.

[7] **Sanjay Ghemawat, Howard Gobioff,Shun-Tak Leung.** *" The Google file system. "*, New York,USA : ACM, 2003 .

[8] **Burrows, Mike.** *" The Chubby lock service for loosely-coupled distributed systems. "*, Berkeley, CA, USA : USENIX Association, 2006. ISBN:1-931971-47-1.

[9] **COMER, Douglas.** *" Ubiquitous B-tree. "*, 2, New York : ACM Computing Surveys, 1979, Bd. 11. ISSN:0360-0300.

[10] **BENTLEY, J. L., AND MCILROY.** *" Data Compression Using Long Common Strings. "*, 1999, In Data Compression, S. 287-295.

[11] **Bloom, Burton H.** *" Space/time trade-offs in hash coding with allowable errors. "*, 1970, Commun. ACM, S. 422-426.

[12] **Rob Pike, Sean Dorward, Robert Griesemer, Sean Quinlan.** *" Interpreting the data: Parallel analysis with Sawzall . "*, 2005, Scientific Programming, S. 227–298.

Next Generation Networks

Dominik Seidel
Betreuer: Heiko Niedermayer
Seminar Future Internet SS2010
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: seideldo@in.tum.de

KURZFASSUNG

Das Internet hat sich heutzutage zu einer etablierten Instanz entwickelt, im privaten sowie im geschäftlichen Umfeld. Jedoch ist aus Benutzersicht ein Wandel der Bedürfnisse entstanden: Weg von klassischen host-to-host Anwendungen hin zu einer auf Namen (des Contents) basierenden Kommunikation. Dem zu Grunde liegt die Tatsache, dass es den Benutzer nicht darum geht WIE oder von WO er den Content bezieht, sondern schlichtweg, dass er den gewünschten Content bekommt. Ein Lösungsansatz für dieses Problem sind die sogenannten Next Generation Networks(NGNs). Diese stellen das oben genannte "content-based" Networking in den Vordergrund, hauptsächlich durch das Verwenden gewisser Naming und Name Resolution Dienste. Für die Zukunft ist noch offen, ob sich diese neuartigen Netzwerke durchsetzen. Sie bieten aber auf jeden Fall gewisse Vorteile und Potentiale, auf die in der Arbeit eingegangen wird!

Schlüsselworte

CCN, DONA, Name Resolution, Naming, NGN

1. EINLEITUNG

Um das Internet als Technologie realisierbar zu machen, war das Domain Name System (DNS) eine der auschlaggebenden Entwicklungen. DNS wurde relativ früh , nämlich in den 1980ern entwickelt, als das Internet noch relativ jung, jedoch bereits schon an Leistungsgrenzen gestoßen war. DNS wurde daraufhin entwickelt, um eine strikte Übersetzung von Host Namen in IP Adressen zu gewährleisten. Ausserdem wurde DNS natürlich von Zeit zu Zeit angepasst und geupdated. Man kann es sich vereinfacht als einen hierarchischen und autonomem Verzeichnisdienst vorstellen, der den kompletten Namensraum verwaltet und in Zonen aufteilt. [4]
Das Problem bei dieser Enwicklung war, dass bereits andere Dienste verankert waren, wie zum Beispiel, dass TCP Sessions an IP Adressen gebunden waren, jedoch nicht an Namen. Somit waren die Möglichkeiten, DNS Namen wirkungsvoll einzusetzen schon im Vornherein durch die gegebene Architektur beschränkt. Man könnte auch sagen, dass sich das Internet auf dieser host-to-host Kommunikation basierend entwickelt hat. Zu dieser Zeit tat das den Ansprüchen auch genüge, nämlich der Kommunikation zwischen einer überschaubaren Anzahl von Hosts. Heute umfasst der Schwerpunkt der Internetnutzung jedoch die Datenübertragung sowie Zugang zu Diensten, wobei dem Benutzer lediglich der Content wichtig ist, und nicht wo dieser liegt bzw. erreicht wird. Letztendlich ist DNS nicht mehr ideal einsetzbar um die Bedürfnisse der Nutzer zu befriedigen und es Bedarf einem Neuansatz auf den hier eingegangen wird. Im Vordergrund der Arbeit steht die sogenannte Data-Oriented Network Architecture (DONA), welche einen Name Resolution Dienst "über" der IP Layer bereitstellen soll. Somit könnten die Mankos der bisherigen Datenübertragung beseitigt und der Zugrif auf Webservices erleichtert werden, sowie Verbesserungen im Bereich der Persistenz, Verfügbarkeit und Authenifikation erreicht werden. Ein Beispiel im Bereich Persistenz wäre hier, dass die sogenannten Broken Links die heutzutage gelegentlich auftreten, dann der Vergangenheit angehören sollten. [3]
In der Arbeit geht es hauptsächlich, um dies noch einmal zu betonen, um die DONA Architektur. Diese ist eine Variante eines NGNs, wobei es natürlich auch noch andere NGN Architekturen gibt, auf die hier nicht näher eingegangen wird. Ferner muss man wissen, dass die VoCCN Implementierung nicht im Zusammenhang mit DONA steht, und eigenständig zu betrachten ist. Deutlich wird dieser Neuansatz auch im Hinblick auf die Architektur, die Gegenstand des nächsten Kapitels ist.

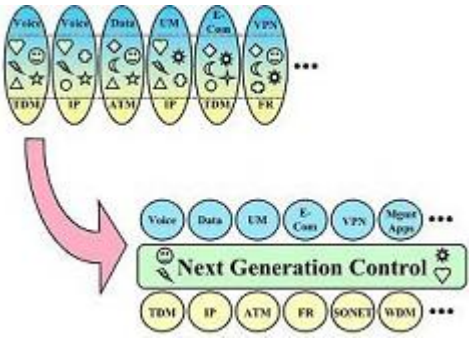


Abbildung 1: Next Generation Control [1]

2. ARCHITEKTUR

Bei der Architektur von Applikationen verwendet man meist je nach Vorliebe/Know-How etc. eine bestimmte zu Grunde liegende Technik, zum Beispiel wird eine Voice Applikation per IP-Protokoll realisiert. Bei einer zugeschnittenen Plattform auf der diese Applikation bzw. eine überschaubare Anzahl an Applikationen läuft, mag dieser Ansatz auch gut funktionieren, wird die Anzahl der Applikationen jedoch höher, kann es hier schnell zu Leistungsdefiziten kommen. Verantwortlich hierfür ist das Problem, dass bei vielen heterogenen Applikationen auch mehrere Verbindungen in ver-

schiedenen Protokollen aufgebaut werden müssen. Allgemein formuliert kommt es zu Einbußen, wenn zu viele unterschiedliche Systeme parallel arbeiten. Genau hier greift auch der Ansatz von Next Generation Networks, nämlich das Bereitstellen einer einheitlichen und flexiblen Steuereinheit (siehe Abbildung 1), die all diese verschiedenen Applikationen und Transportarten verbindet und verwaltet.[1]

2.1 VOIP Vergleich

Um aufzuzeigen, auf welche Punkte es bei der Architektur eines solchen Systems ankommt, kann man sich dies anhand einer einfachen Verbindung per VoiP zwischen zwei Benutzern klar machen. Abbildung 2 zeigt eine standardmäßige VoiP Verbindung:
Wenn Alice und Bob telefonieren wollen, wird zunächst eine Audio Verbindung per Session Initialiation Protocol (SIP) hergestellt. Durchgeführt wird dies typischerweise über einen Proxyserver, da die VoiP Endpunkte oft mobil sind oder dynamische IP Adressen verwenden. Akzeptiert Bob die Gesprächseinladung, so wird ein direkter "media path" zwischen beiden Endpunkten hergestellt. Hier zeigt sich das Dilemma, das in Verbindung mit der "alten" Architektur steht: Alice hat das simple Ziel direkt mit Bob zu reden. Das Netzwerk braucht um eine Verbindung herzustellen, jedoch den komplizierten Umweg aus Abbildung 2. Der Grund dafür ist, dass das Netwerk nicht direkt Bob's Telefon adressieren kann sondern lediglich seine IP Adresse!
Schaut man sich eine Realisierung der VoiP-Kommunikation

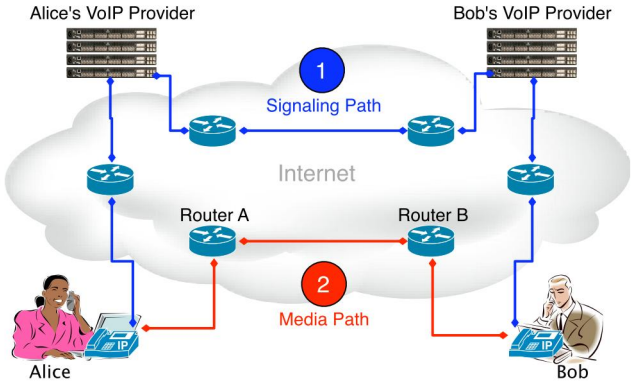


Abbildung 2: Klassische VoiP Kommunikation [2]

per Content-Oriented Network Architektur in Abbildung 3 an, sieht man schnell dessen Vorzüge: Hier fällt nämlich die komplette Übersetzung durch eine Middleware weg, sprich der oben genannte Umweg über Proxies und ein SIP. Man hat hier nurnoch eine einsträngige Verbindung, die Signaling und Media path in Einem ist. Natürlich bedarf dieser Verbindungstyp noch zusätzlichen Anforderungen, zum Beispiel muss der Angerufene einen "service contact point" bereitstellen, sozusagen einen Andockpunkt für das Telefonat.
Zusätzlich sind noch andere Voraussetzungen nötig, auf die in dieser Arbeit jedoch nicht näher eingegangen wird.
Wichtig ist aber, dass man schon anhand der zwei Grafiken

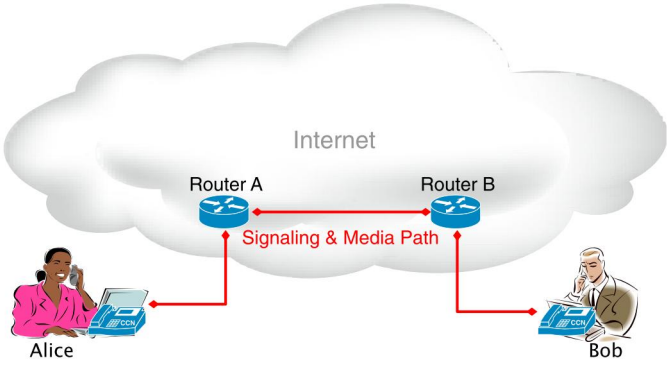


Abbildung 3: Voice-over-CCN Kommunikation[2]

sehen kann, dass hier eine Verringerung der Komplexität erzielt werden kann, abhängig von der verwendeten Architektur.[2]

2.2 Naming

Einen wichtigen Punkt der DONA Architektur ist die Zusammensetzung von Namen des Systems. Zentraler Punkt sind die sogenannten "principals", die Ersteller des Contents: Jeder principal lässt sich einem public-private Schlüsselpaar zuordnen, und jedes Datum, jeder Service oder Entity ist einem principal zugeordnet. Somit haben die Namen allgemein die Struktur P:L, wobei P der Hashwert des public keys (des principals) ist, und L die Benennung durch den principal. Als Resultat ergibt sich eine Namenstruktur die dafür sorgt, dass alle Namen einzigartig sind.
Ein weiterer Faktor ist das Besitzrecht der principals. Um genauer zu sein kann der principal festlegen, welchen Hosts er Zugriff auf die von ihm generierten Daten der Form P:L gewährt. Vorteil dabei ist auch, dass die Integrität der Daten damit auch garantiert/geprüft werden kann. Eine Verifizierung kann mithilfe der einzigartigen Namen und anhand der Kombination aus den eigentlich Daten, dem public key des principals und der Signatur erfolgen. Des weiteren kann durch einen Check, ob der public key denn wirklich zu P passt, Authentifikation garantiert werden. Persistenz wird dadurch erreicht, dass die Daten nicht an einen gewissen Ort gebunden sind, sondern überall bereitgestellt werden können.
Was man jedoch auch beachten muss, ist dass durch diese Technik auch Probleme entstehen, die es zu lösen gilt. Das zentrale Problem ist etwa, wie sich die Benutzer diese langen, komplex zusammengestellten Namen merken sollen. Hier schafft man sich Abhilfe, indem man externe Mechanismen verwendet, bei denen sich der Benutzer im Bezug auf die Sicherheit bzw. Vertrauenswürdigkeit seiner Daten sicher sein kann. Beispiel wäre eine Suchmaschine, die die komplexen Namen dann in menschenlesbare Namen umwandelt.[3]
Nun stellt sich die Frage, wie man sich sicher sein kann, dass diese Namen auch auf die gewünschten Ziele führen. Dies ist Aufgabe der Name Resolution, was im nächsten Abschnitt erläutert wird.

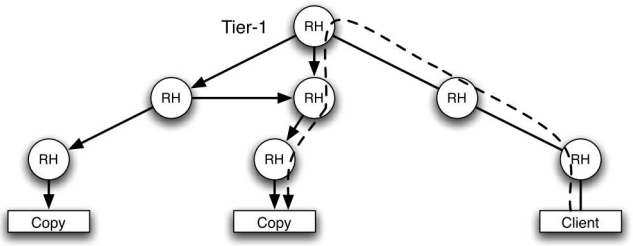


Abbildung 4: REGISTER = ganzen Pfeile und FIND = gestrichelten Pfeile[3]

2.3 Name Resolution

Nun gilt es zu erläutern wie DONA die Namen auf die gewünschten Hosts mappt. Ziel dieses Dienstes ist es eine hohe Verfügbarkeit der Daten zu erzielen, indem nahegelegene Kopien der Datei gefunden werden können, sowie die Fehleranfälligkeit minimal gehalten wird. DONA benutzt hierfür sogenannte Resolution handlers (RHs), die eine neue Art von Netzwerkentität darstellen. Jede Domain oder Verwaltungseinheit verfügt dann über eine spezifische RH. Denkbar ist auch das Untergliedern von sozialen Strukturen mit Hilfe der RHs. Zum Beispiel könnte an der TU München jeder Lehrstuhl einen eigenen RH haben, und dieser könnten dann für die Mitarbeiter am Lehrstuhl intern andere Rechte besitzen, im Gegensatz zu Lehrstuhl externen Personen. Realisiert wird diese Technik, indem jeder Client seinen lokalen RH kennt und diesem zugeordnet ist.

Desweiteren stellt DONA zwei Basisbefehle zur Verfügung, nämlich einmal FIND(P:L) und REGISTER(P:L), die in diesem Abschnitt erläutert werden. FIND hat den Zweck, solche Daten mit dem Namen P:L zu finden, und die Rhs sorgen dafür, dass eine nahegelegene Kopie davon gefunden wird. Im Gegensatz dazu sorgt REGISTER dafür, das Daten sozusagen für andere auffindbar sind, und somit durch FIND Anweisungen gefunden werden können.

Ein FIND(P:L) wir abgehandelt, wie in Abbildung 4 teils ersichtlich, indem zuerst geschaut wird ob unter den RHs eine den gesuchten Namen registriert hat. Ist dies der Fall, wird der FIND zum nächsten RH geleitet, so lange bis der Pfad bis zu dem RH mit der gewünschten Kopie erreicht wird. Im Bezug auf unser Beispiel wandert der FIND erst einmal bis zur Wurzel, und daraufhin bis zum Blatt mit der gewünschten Kopie. Zusätzlich initiieren die FIND Befehle auch den eigentlichen Austausch/Transport der Daten. Nur der Austausch der eigentlichen Datenpakete erfolgt per gewohnter IP Routing Technik. Das hat den Vorteil, dass an der bestehenden IP Infrastruktur keine Änderungen vorgenommen werden müssen.

Bei den REGISTER(P:L) Befehlen ist das Verhalten ein wenig anders: Wenn ein RH einen solchen erhält, wird dieser lediglich nach oben weitergereicht, wenn es eine nähere Kopie enthält (im Vergleich zur bestehenden "Struktur") oder wenn es sich um ein neues Objekt handelt. Ausserdem bedarf es einer Authentifikation, da garantiert werden muss, dass es sich bei P um eine vertrauenswürdige Quelle handelt. Der Client muss sich deshalb in der Regel mit P's private key anmelden.[3]

3. FUNKTIONEN

Die zu Grunde liegenden Architekturmerkmale wie die Name Resolution spiegeln sich natürlich auch in der Funktionalität von DONA wieder. Grundsätzlich lassen sich diese in vier Bereiche aufteilen, die in der Arbeit nun erläutert werden.

Erster ist die Selektion des Servers (bzw. mehrerer Server). Will ein Server einen Service mit dem Namen P:L registrieren, so benötigt er erst einmal die Authorisierung von einem principal P. Ist das erfüllt, dann wird der Service einfach beim lokalen RH angemeldet, und ist ab sofort dann auch für FIND Anweisungen zu finden. Natürlich ist es so, dass immer der näheste Server gewählt wird bei identischen Services. Durch diesen Mechanismus wäre es auch kein Problem zum Beispiel P2P Applikationen wie BitTorrent zu realisieren. Hier würde man einfach einzigartige Namen verwenden, mit denen der Principal die jeweilige P2P Instanz identifizieren kann. Probleme würden auch nicht auftreten, wenn wie bei BitTorrent üblich, die Dateien in einzelne, kleine Teile gesplittet werden. In diesem Fall würde man sich einfach mit einem Index Abhilfe verschaffen, der die einzelnen Teilnamen dem Dateinamen zuordnet! Der Client könnte somit auch sehr simpel nur einzelne Teildateien bekommen.

Zweite wichtige Funktionalität ist der "mobile" Charakter. Dies soll heißen, dass ein Host sich einfach an einer Location abmelden kann und beim Wiederverbinden die neue Location anmeldet. Somit würden FINDs dann auch direkt an diese neue Location geleitet werden, sobald die Registrierung erfolgt ist. Ähnlich läuft dies beim sogenannten Multihoming ab, sprich wenn man mehrere ISPs besitzt: Automatisch wird dann ein REGISTER an alle Provider übermittelt. Gleichzeitig können FINDs dann auch multiple Bezugsquellen benutzen.

Nächster Punkt ist die "Session Initialition". Eine zentrale Rolle spielen hierbei das sogenannte Session Initiation Protocol (SIP), welches eng mit den grundlegenden DONA Funktionen (FIND und REGISTER) verknüpft ist. Typischerweise erfolgt dann der Aufbau einer Session zwischen zwei Partnern wie folgt: Zuerst schickt ein SIP Agent eine SIP INVITE Nachricht an den Sessionpartner. Das SIP Proxy leitet diese Nachricht dann an die Location des Zielagents weiter und wenn dieser antwortet, startet die Session. Um die Location auch immer auf dem aktuellen Stand zu halten, registrieren SIP Agents kontinuierlich ihre Position. Im Zusammenhang mit FIND/REGISTER kann man sagen, dass eine SIP Invite Nachricht vergleichbar mit einem FIND ist, sowie eine SIP und DONA REGISTER Nachrichten ziemlich ähnlich sind. Anzumerken ist noch, dass wie schon einmal erwähnt, der eigentliche Datentransfer standardgemäß per IP Protokoll erfolgt.

Der letzte Punkt ist das "Multicast State Establishment". Diese Funktion beinhaltet Die Möglichkeit über Domaingrenzen hinaus anderen Gruppen ohne Problemen beizutreten. Zum besseren Verständnis nehme man an, es gebe einen Domainrouter mit lokalen Mitgliedern in einer Gruppe X. Nun muss aber gewährleistet sein, dass auch Gruppenmitgliedern die aus anderen Domains stammen, der Zugriff möglich ist bzw. mit diesen kommuniziert werden kann. Dies ist Sinn und Zweck dieser Funktion, die per "intradomain multicast protocol" solch ein Überschreiten der lokalen Domaingrenzen ermöglicht. Auch im Hinblick auf die Gruppennnamen erkennt man wieder die typische Struktur: Gruppen haben Namen der Form P:G, wobei P wieder der principal ist,

also der Gruppengründer und G der eigentliche Gruppenname. Damit ist ebenfalls wieder die Einzigartigkeit der Gruppennnamen garantiert.[3]
Dies waren die Grundfunktionalitäten, wobei es noch zahlreiche spezifischere Erweiterungen für DONA gibt bzw. in Arbeit sind. Auf diese wird im Rahmen dieser Arbeit jedoch nicht eingegangen, da dies eher Feinheiten sind die für ein Grundverständnis für NGN's/DONA nicht noetig sind. Wichtig ist aber der nächste Punkt, nämlich der Sicherheitsaspekt in einem solchen System.

4. SICHERHEIT

4.1 Beispiel VoCCN Sicherheit

Um ein Beispiel mit Praxisbezug zu haben, kommen wir zunächst noch einmal auf das VoCCN Beispiel (Abbildung 3) zurück. Hier wurde Media Path und Signaling Path seperat gesichert.
Bei Beiden benutzt man die Authentifizierung der User per Key-paar. Eine Verbindung kommt also nur zu Stande wenn die entsprechenden Keys verifiziert werden. Verantwortlich für die Verteilung dieser ist ebenfalls das CCN Netzwerk. Zusätzlich wurden im Media Path alle Gespräche per Secure Real-Time Transportation Protokoll geführt. Dieses sorgt grob gesagt für eine Verschlüsselung, Authentifizierung und Integrität des Nachrichtenaustauschs.
Die Sicherheit des Signaling Path wird garantiert, indem ein einfaches Verschlüsselungs- und Authentifizierungsschema implementiert ist. Genauer gesagt wird die SIP Invite Message verschlüsselt und benutzt einen zufällig generierten symmetrischen Schlüssel auf Seiten des Angerufenen. Der Anrufer würde diesen symmetrischen Key dann seinerseits per public key seines Gegenübers entschlüsseln.
Als Resultat hat man laut den Entwicklern dann eine deutlich sicherere Variante im Vergleich zur herkömmlichen VoiP Kommunikation. Diese ist nämlich oft komplett unverschlüsselt und ohne Authentifizierung![2]

4.2 Sicherheitsbelange

Der erste kritische Punkt sind denial-of-service Attacken die RHs, Server oder Clients "überfluten" könnten, was jedoch per IP-level Mechanismen in der Regel verhindert wird. Größeres Problem stellen schon Angriffe dar, die zum Ziel haben, die Ressourcen einer RH zu limitieren/einzuschränken. Hier müsste man sich dann mit den ISPs verständigen. Genauer gesagt müssten diese ein Limit an FINDs und REGISTERs pro Zeitperiode festsetzen.
Betrachtet man den Austausch der RHs untereinander, ist man schon zum Großteil dadurch abgesichert, dass die RHs untereinander ihre public keys austauschen und verifizieren. Dadurch kann garantiert werden, dass das gesendete Paket auch von der gewünschten RH stammt. Eine böshafte RH ist dadurch jedoch noch nicht abgewehrt. Diese könnte zum Beispiel erhaltene REGISTER Befehle schlicht und einfach verweigern, insbesondere das Weiterleiten an die nächsten RHs. Noch schlimmer wäre der Fall, wenn eine bösartige RH die REGISTERs anderer RHs abhört und dessen Daten sammelt. Um dies zu verhindern wird weitergeleiteten REGISTERs stehts der public key der nächsten RH angefügt. Ein Dilemma ware natürlich auch das Verweigern des Service einer RH an den oder die Clients. Dieses Problem kann aber behoben werden, indem der Client diese RH als fehlerhaft erkennt und automatisch eine andere Kopie der Daten ansteuert.
Im Allgemeinen kann man sagen, dass die RHs in der Regel von den Internet Service Providern (ISPs) verwaltet werden und Kunden dafür zahlen. Folglich wird es im Interesse aller liegen, wenn bei Problemen so schnell wie möglich eine Lösung gefunden wird. Vor allem die ISP werden es sich nicht leisten können, Sicherheitslücken zuzulassen.[3]

5. IMPLEMENTIERUNG

Im Bereich der Implementierung gilt anzumerken, dass die meisten Testsysteme nur im kleinen Umfang gearbeitet haben. Dies liegt daran, dass meist nur Prototypen implementiert wurden und logischerweise ein System im Umfang des eines ISPs umzusetzen, im Rahmen von Tests nicht möglich war. Man sieht jedoch gut, wie die einzelnen Module eines solchen Systems agieren.
Als erstes benötigt man ein Router Modul. Dieses handelt empfangene REGISTERs und FINDs ab und leitet diese bei Bedarf weiter. Es beinhaltet auch das Registration Table, welches eben FINDs gezielt an andere RHs weiterleiten kann. Aussderdem überwacht das Router Modul den Status der angeschlossenen Nachbarn. Mit diesen Nachbar-RHs werden auch REGISTER Daten ausgetauscht, sodass jede RH immer auf dem neusten Stand ist.
Ein weiteres Modul ist das Caching Modul. Dieses ermöglicht den Zugang zu dem lokalen Cache von Daten. Darauf können dann gewisse Applikationsmodule zugreifen, zum Beispiel eine P2P Applikation. Wird der Cache abgefragt erfolgt dies in zwei Schritten: Erst wird ermittelt ob das gesuchte Objekt im Cache ist. Ist dieser Vorgang erfolgreich, wird auf das Objekt regulär per Dateisystem zugegriffen.
Letztes Modul sind die Application Modules, die oben schon genannt wurden. Sehr üblich sind Dinge wie HTTP, SIP,P2P Applikationen oder RSS.
Allgemein hatte man bei dem implementierten DONA Prototyp an die 17.000 Zeilen Programmcode. Aber wie gesagt konnte man noch keine Aussagen über die Skalierbarkeit machen, da dafür das System noch zu wenig Umfang hatte.[3]

5.1 Performance

Dier Performance konnte man jedoch schon ganz gut bei der VoCCN Implementierung vergleichen. Zunächst war die Gesprächsqualität laut der Verfasser recht gut. Getestet wurde zwischen zwei schnellen Workstations mit 100 Mb/s oder im Wechsel mit 1Gb/s.
Die genauen Testergebnisse lassen sich ganz gut in Abbildung 5 ablesen. Hier wurde jeweils ein 10 minütiges Gespräch getestet, einmal per VoCNN Implementierung (gestrichelte Linie) und einmal über die "'normale"' Variante mittels UDP. Abweichend von den erwarteten Resultaten hat VoCCN deutlich weniger Pakete unter oder bei der erwarteten Intervallzeit. Im hohen Intervallbereich sind lediglich wenige Pakete transferiert worden. Alles in allem sind bei keiner der beiden Varianten Pakete verloren gegangen![2]

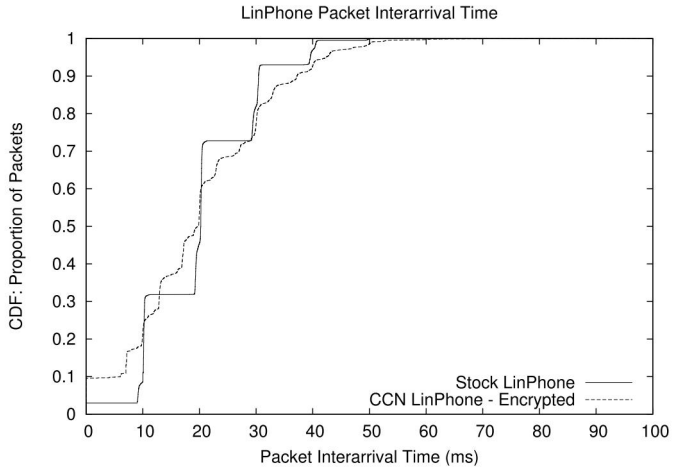


Abbildung 5: Verteilung der Paketintervalle (kumultativ) bei einem 10 min Anruf[2]

6. REALISIERBARKEIT

Da nun alle wesentlichen Fakten im Bezug auf DONA geklärt sind, stellt sich die Frage ob in der Realität ein solches System überhaupt umsetzbar ist oder ob vieleicht die Kosten ein solches zu betreiben den wirtschaftlichen Rahmen eines ISPs sprengen wuerden. Angemerkt sei, dass im Bereich einer "'durchschnittlichen"' RH keine grossen Leistungsengpässe zu erwarten sind. Grund dafür ist, dass eine RH ja wie gesagt baumartig arrangiert ist, sprich sie muss nur FINDs und REGISTERs von RHs auf gleicher Ebene oder darunter abhandeln. Lediglich die Wurzel im Baum, daher die Tier-1 RH, welche seitens des Providers verwaltet wird muss sogesehen das volle Spektrum der registrierten Datennamen wissen

Schätzt man zunächst den Platz der zum Speichern aller (öffentlichen) Webseitennamen nötig wäre, würden wir auf ca. 10 hoch 11 Seiten kommen. Geht man nun von 42 byte pro Eintrag aus, kommt man auf lediglich 4 TB benötigten Speicher. Da heutzutage die Preise für Speicher dieses Umfangs sehr erschwinglich sind, stellt dies kein Problem dar.

Geht man ferner von einer durchschnittlichen Lebensdauer von zwei Wochen pro REGISTER aus, muss eine Tier-1 RH ungefähr 83000 Registrierungsnachrichten pro Sekunde verarbeiten können. Umgerechnet entspricht dies 680 Mbps, was für ein RH eines ISPs kein Problem darstellen darf. Was jedoch weitaus aufwendiger ist als der blose Register-Befehl, sind die komplizierten kryptograpischen Prozesse die gleichzeitig anfallen. Hochgerechnet wäre für diese Arbeit die Leistung von ca. 40 3 GHz CPUs nötig. Trotz allem ist diese Anzahl doch durchaus im Rahmen und stellt für einen ISP kein Problem dar.

Auf Seiten der FINDs kann man davon ausgehen, dass diese ungefähr die selbe Last verursachen wie eine gewöhnliche HTTP Anfrage. Man kann annehmen, dass eine voll ausgelastete 1 Gbps verbindung etwa 20.000 FINDs pro Sekunde verursacht. Rechnet man bei einem FIND 150 byte Volumen, wären das nur 24Mbps die nötig sind, sprich 2,4 Prozent der Leitung. Nehme man nun des weiteren an, dass ein CPU um die 40.0000 FINDs pro Sekunde verarbeiten kann, so können 500 PCs mit jeweils 8 GB RAM summa summarum eine Last von 1 Tbps verkraften. Dies wäre schätzungweise auch ein sicherer Wert der noch Puffer zulässt. Wohlgemerkt wäre mit diesen 500 PCs die komplette Tier-1 eines ISPs abgedeckt, was durchaus im Rahmen einer wirtschaftlichen Tätigkeit liegt.[3]

7. ZUSAMMENFASSUNG

Betrachtet man nun diese NGNs im Hinblick auf die Zukunft, haben diese durchaus Potential Nachfolger der herkömmlichen Netwerke mit DNS etc. zu werden, da sie das was den Kunden und Benutzer am meisten interessiert, den Content, in den Mittelpunkt stellen. Dieser kann dann durchaus effizient und skalierbar bereitgestellt werden. Auch klassische Dienste wie telefonieren, e-mailen oder surfen kann per CCN effektiv bereitgestellt werden. Das Ergebnis muss nicht ungedingt besser sein als das der herkömmlichen IP-Struktur, entscheidender Faktor ist aber: Die Vereinfachung in der Architektur, Implementierung und Konfiguration ist der ausschlaggebende Wegweiser. Darüber hinaus spielt der Sicherheitsaspekt, im Hinblick auf Datenklau Skandale etc., zur heutigen Zeit eine immer wichtigere Rolle. Da dieser bekanntlich (siehe Abschnitt 3) wesentlich höher als bei den jetzigen Netzwerken ist, besteht auch hier hohes Potential. Vor allem werden keine Zwischenstationen mehr, wie z.B. Proxies benötigt.[2]

Allerdings ist ein wesentlicher Faktor, der wahrscheinlich für den Erfolg oder Misserfolg verantwortlich sein wird, noch fraglich: Nämlich ob diese Technologie die Aufmerksamkeit der Telekommunikationsbranche und deren Unternehmen auf sich ziehen kann. Denn ohne eine weitflächige NGN Umwelt wäre dessen Durchsetzung auf dem Markt wohl eher unwahrscheinlich. Zusätzlich werden erst durch eine breite Wahrnehmung/Entwicklung der Industrie die wirklichen Voraussetzungen für NGNs bekannt werden. Unternehmen sollten deswegen die NGN Technologie als Differenzierungsstrategie sehen um sich von der Konkurrenz abzuheben![1]

8. LITERATUR

[1] J. Crimi. Next generation network (ngn) services. *Telcordia Technologies*, 2009.

[2] V. Jacobson, D. Smetters, and N. Briggs. Acm rearch '09. In *VoCCN: Voice Over Content-Centric Networks*, Dezember 2009.

[3] T. Koponen, M. Chawla, and B.Chun. Sigcomm'07. In *A Data-Oriented (and Beyond) Network Architecture*, August 2007.

[4] P. Vixie. Dns complexity. *Queue*, 5(3):24–29, April 2007.

High Speed Network Monitoring

Leonhard Uden
Betreuer: Nathan Evans
Seminar Future Internet SS2010
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: uden@in.tum.de

ABSTRACT

Network monitoring is an important activity to ensure a smooth working of IP networks. Operators of networks utilize it to measure the traffic, plan new investments or perform intrusion detection. Since the bandwidth of network links increases much faster than the processing power, centralized architectures are no longer capable to capture and monitor the amount of traffic or depend on high performance hardware.

Distributed architectures which split up the traffic to several capturing nodes are a promising solution to this problem. They provide scalability and consist of modified standard PCs. This paper will present a proposal of such an architecture and compare its design to other approaches. Different dispatch methods will be compared and analyzed. The performance of distributed designs outperforms centralized ones when they use similar hardware. Issues still exist, since load balancing and comprehensive monitoring is difficult to achieve at the same time.

For now centralized architectures using high performance hardware are the better solution for network operators. Web development and different growth rates of bandwidth and processing power lead the development of high speed monitoring towards a distributed solution.

Keywords

Network Monitoring, Distributed Architecture, High Speed Networks.

1. INTRODUCTION

In recent times the amount of data transferred by IP networks constantly increased. This has happened by providing new services which are often free of cost. Older analog technologies are getting displaced by newer ones using IP networking. Skype and youtube are the most famous examples of this trend.

In order to guarantee a certain quality of service, detect malicious traffic and plan new investments, network operators have to monitor and analyze traffic. They are challenged by this development, since the growth in amount of data transferred is much higher than the growth of processing and memory speed. A forecast of Cisco expects the IP traffic to be five times higher in 2013 than in 2008, which means a yearly growth rate of 40 percent [2]. When it comes to IDS (Intrusion Detection Systems) the increasing possibilities of malicious traffic extend the process of scanning a packet. Nowadays it is common to use 10 Gigabits per second Ethernet links in bigger networks. Such a fully loaded link transfers a packet in less than 100 nanoseconds, by decreasing packet-sizes, therefore increasing packet-number, the processing time for each packet decreases. That means a capturing device has only nanoseconds to capture a packet. Also the time needed for analyzing packets has to be considered. There are different ways to deal with this challenge.

The most obvious one would be to use special high performance capturing devices which are able to capture at Gigabits per second Ethernet transfer rates. The dedicated capturing cards of Endance[4] provide capturing at high transfer rates. These cards guarantee to capture 100 percent of the packets, are able to distribute the traffic to different memory buffers, perform time stamping and reconstruct a replication of traffic. Drawbacks are higher prices and less flexibility compared to other approaches.

The next option is to lower the capture rate by using sampling. This method is popular today because it is sometimes already implemented into routers and does not have a need for high performance hardware. Here the questions are how high the sampling rate should be and if one can make founded conclusions to the rest of the packets which are not captured. Research has shown that sampling is accurate in computing the total amount of traffic, since the packet-size does not vary too much. Sampling has a lack of accuracy when it comes to flow count, smaller flows could be missed entirely. That often causes anomalies to stay undetected [1]. Recent research on sampling methods has found ways to increase the rate of detection, but this won't be handled in this paper, overviews are given in [14].

This paper will describe distributed architectures. These architectures try to split the traffic and allocate it to different capturing or analyzing devices, in order to achieve a high capture rate and avoid sampling. There have been several projects on this topic recently, like the IDS architecture described in [12] or the DaSahit project (Distributed and Scalable Architecture for High-Speed IP Traffic Analysis) [3]. Different rules are applied to split the traffic: to sort the packets by port, IP address, or by using some sort of algorithm.

DiCAP(Distributed Packet Capturing Architecture for High-Speed Network Links) [9] will be used as an example for such an architecture. It is especially designed to capture traffic, not to analyze the packets, which means it only retrieves the data of a network link and does not scan the data. The aim is a scalable architecture that can be flexibly extended. A major advantage of this method is the possibility to split the traffic into such small amounts that it can be processed by inexpensive standard PCs using Linux.

The paper is organized as follows. In section 2 DiCAP will be presented and reviewed. Section 3 compares different approaches of creating a distributed architecture for monitoring or analyzing IP traffic. The last section, number 4, concludes the findings of this paper.

2. DiCAP

Uncoupling packet capturing from its limitations is a major motivation for distributed architectures. The authors of DiCAP present a proposal that is able to handle all kinds of protocols, is not dependent on high performance hardware neither lacks accuracy like centralized sampling methods. It is an easy scalable architecture using standard hardware.

Distributing the workload of capturing packets of high speed links to multiple devices is the main idea of distributed architectures. In DiCAP those devices are called capture nodes, are based on regular PCs with a modified NIC driver. They are connected with a router called mirroring device, a standard PC which coordinates them and eventually with an external analyzing device which is not considered in this paper. The distribution of workload is achieved by forwarding the traffic to each capture node and using a kind of sampling on each that only allocates a unique selection of the packets to each node. How this is done will be explained and analyzed in the next sections, which are all based on the paper written by Morariu and Stiller [9].

2.1 Architecture

On a network link a mirroring device is installed, its task is to mirror the passing traffic and to forward the copied packets on to the capture nodes. A drawback is that if mirroring fails, capturing and analyzing fails. Security dependent networks might also use the mirroring device as a "gate" on the main link, so if the device fails, mirroring is no longer possible, but also no traffic is able to pass. Via multicast the packets are sent to the capture nodes.

The cluster of capture nodes is organized by a node coordinator. It tells each capture node how to decide which packets to capture and which to discard. To avoid a breakdown of the whole system, because of a malfunction of the node coordinator, other node coordinators could be installed. They would be synchronized every n seconds and stay inactive, until the active coordinator fails and another one is chosen as a replacement.

Every capture node has a unique nodeID that is used for identification when the topology is set at the node coordinator. A capture node has two interfaces, a passive one just for retrieving the mirrored traffic and an active one for communicating with the current active node coordinator and the packet data analyzer(s). When a packet arrives at the capture node, the DiCAP module decides whether to drop the packet or to capture it. The DiCAP module is a software extension of the NIC driver (see Section 2.1.3). If the packets should be captured is decided by using a function. The authors propose the use of a hash based selection or round robin selection. Which one to be used is defined by the node coordinator.

There are two different solutions existing. The first one is called distributed capture mode (see Figure 1). In this mode the DiCAP module stores only the first 54 bytes in a buffer, so only the packet headers are stored. The payload is dropped, this leads to a lack of data integrity. If the buffer is full a UDP message is sent to the packet data analyzer(s).

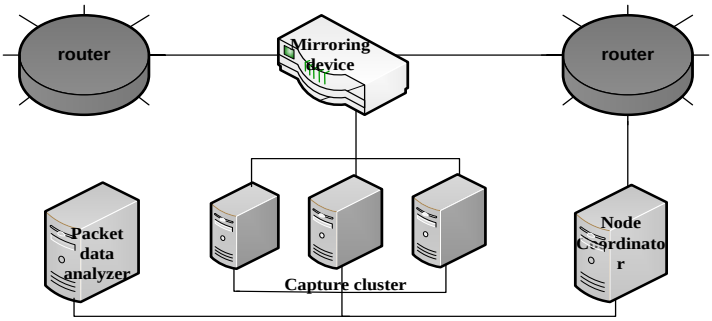


Figure 1: Distributed Capture Mode

How the captured packets are distributed may be discussed in later sections, since this is only a description of the monitoring architecture.

The other mode is called distribution mode (see Figure 2). Here the DiCAP module is not capturing the packets, but forwards the packets it decided to keep to an external capture tool. The prototype uses libpcap[7] for this task. So DiCAP is only a distributor in this mode.

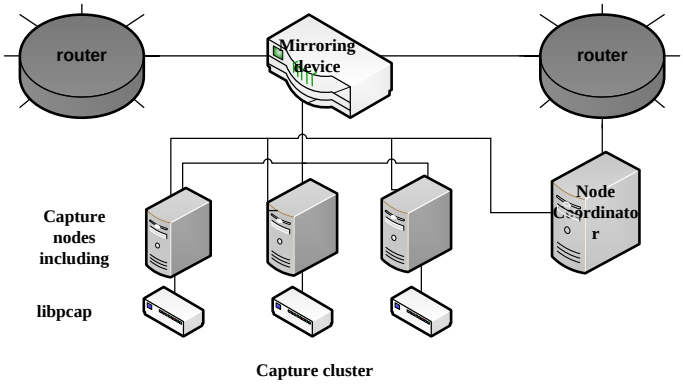


Figure 2: Distribution Mode

2.1.1 Communication

In order to achieve correct traffic distribution without sampling effects, it is necessary that every capture node is aware which part of the traffic it should capture. Since the distribution can change when new nodes enter or leave the cluster, the node coordinator has to be informed about changes and update the topology regularly. A node is an active member as long as it regularly sends heartbeat messages.

This process is handled by sending control messages via the active interface of the nodes. The generic control messages are specially defined for this task. They use seven different so called AVPs (Attribute-Value-Pairs) to exchange information between the coordinator and the capture nodes. The

- nodeID which identifies the capture node
- Coordinator IP/Port which tells the capture nodes which IP address/port is used by the coordinator
- Analyzer IP/Port which tells the capture nodes where they should forward the captured data
- Selection Type which tells the capture nodes how to select the packets to be captured

- Validity Start which specifies when new topologies should be coming into effect

A selection of these values is sent with the different messages exchanged. The three message types are called join, accept and topology update. In a join message a node who wants to join the cluster sends a message with his nodeID to the coordinator. An accept message confirms the joining. By sending a topology update message the coordinator defines the topology of the cluster.

2.1.2 Dispatch Method

As mentioned before the authors chose two different distribution solutions. One of them is a round robin selection mode, in this mode every capture node has a definite position (P_a) in the total amount of active capture nodes (N). Each node has a packet counter C, which is always set to zero when a topology update happens. A node captures a packet if:

$$C \bmod N = P_a$$

else the packets are dropped. This selection mode allocates every capture node a data amount of 1/N of the total data, which means a perfectly balanced workload. This way each capture node processes sampling but all together they capture the whole traffic. The mode only works, if a perfect synchronization of the individual nodes is given. The different counters always have to be equal and the traffic received always in the same order, else traffic will be missed or captured twice. All nodes have to get all update messages and work always properly to ensure an exhaustive capturing process.

Another method utilizes a hash function. The challenge is to find an appropriate hash function which is easy to calculate and has well balanced outcomes. Also a well distributed value as base for input for the hash function is needed. With the position of the capture node (P_a), the total number of active nodes (N) and an input value (I) given, packets are captured if:

$$\text{hash}(I) \bmod N = P_a$$

else the packets are dropped. The authors discovered that the identification field in the IP header could meet the expectations of a well balanced input value. A header field that identifies the fragments of a packet and is used for reassembling does not include information about flow membership, additionally in IPv6 the field is only existent in an extension header. To use a header field in combination with a hash function might be a good solution, but it is hard to find an appropriate header field and a hash function which meets the criteria of load balancing and flow preserving.

If the main criteria for monitoring, is just to measure the total amount of packets and perfect load balancing, the round robin selection mode should be preferred. It provides a perfectly balanced distribution of traffic and incrementing a counter is a cheap operation. It should not be a problem to keep the traffic in order since only short Ethernet links are used. A concern of the round robin selection mode is the synchronization of the individual counters. If it is possible to synchronize the individual capture nodes when a topology update occurs is uncertain. If a node captures packets of a 10 Gbps network link, it might has to capture a packet every 40 ns, therefore the counter of the nodes have to be accurate to nanoseconds. If the counters are not synchronous, different nodes capture the same packets and therefore some packets will not be captured because they are not in the scope of any node. A solution to this problem might be to interrupt the traffic forwarded by the router for a short period of time, in order that the capture nodes have more time to reset their counters. Only if the synchronization is ensured, the total amount of packets is captured. Topology updates will not happen very often once the architecture is established, therefore the interruptions of the mirrored traffic are negligible.

For comprehensive monitoring it is necessary that flows are detected and captured at the same device. This is important for flow path analysis and IDSs. It is not reasonable to split flows and try to reassemble them later on. That would cause an expensive processes and very good communication between the different nodes is necessary. A large distribution process to exchange packets between the capturing devices and analysis devices would be the outcome. A method which sorts the packets by their attributes in the header, eventually combined with a hash function, is necessary for an effective comprehensive monitoring.

2.1.3 Capture Node

Every capture node could be a standard PC, the only specification is that it has two network interfaces. Capturing is organized by a so called DiCAP module that is also implemented in the coordinator nodes and the packet data analyzer (if they exist). This module is a configuration of the driver of the NIC (Network Interface Card). Its task is to decide whether to capture a packet, forward it or drop it, before the kernel allocates memory to the packet. The DiCAP module consists of a management unit, a packet processor and a packet data forwarder (see Figure 3).

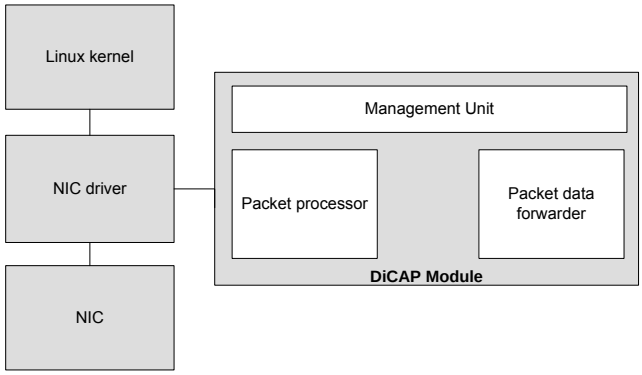


Figure 3: DiCAP Module

Communication with the node coordinator and determining the behavior of the module is the task of the management unit. It has information about the addresses and which network topology is used. The packet processor only handles packets received in the passive interface, the others are sent to the kernel. It has a different behavior for each mode. In distributed capture mode the packets received on the passive interface are sent to the packet forwarder if they are in the responsibility of the capture node, if not they are just dropped. The monitored packets are never processed by the kernel in this mode and always dropped by the NIC driver. Packets to be monitored are delivered to the packet forwarder, which stores the headers in a buffer that it sends to a packet analyzer via UDP when the buffer is full.

While using the distribution mode, the packet forwarder is deactivated, since the captured packets are locally processed by libpcap. Here the packets that should be captured are forwarded to

the kernel and other packets are dropped by the NIC driver. In this mode the task of the DiCAP module is not to capture the packets but to determine which ones to analyze at which analyzing device.

2.2 Review

As seen for the purpose of just capturing packets at high rates DiCAP is a well balanced architecture. But if the project should ever be used further there are some criteria that hinder use in a more complicated environment.

The implementation of the design is good when it comes to scalability. To extend the capture cluster a pretty automatic procedure was created. With the creation of a new protocol to communicate between the coordinator and the capture nodes it is easy handled for a node to join the cluster. One join message, one accept message and a topology update suffices to extend the cluster. Another quality is the opportunity of combining the system with other software, as is done with libpcap in the distribution mode.

Fault tolerance is achieved by the system of multiple coordinator nodes and heartbeat-messaging of the capture nodes. If a capture node does not send its heartbeat message for a determined time, the node coordinator erases it from the list of active nodes and performs a topology update automatically. If a node coordinator breaks down it is replaced by another one which was synchronized before. The weakness is the mirroring device. It cannot be easily replaced and only exists once in the prototype.

Performance tests by the authors have shown that a single DiCAP device, used on a single node in distributed capture mode where only the first 54 bytes of each packet are captured, outperforms other devices using libpcap or libpcap-PFRING [11]. At high packet rates of 620Kpps(thousand packets per second) where the libpcap and libpcap-PFRING devices have a loss rate of 93 percent and 96 percent, the DiCAP device still has a packet loss rate of 0 percent. When used in distribution mode in combination with libpcap a performance increase by increasing number of nodes is remarkable. It is also reasonable to use two different devices for communication and monitoring, since communication packets always have to be processed and some monitoring packets are dropped.

The implementation of the DiCAP module prevents the kernel from allocating memory to packets which are not captured. Since first the module decides whether to drop a packet or not and second memory is allocated to packets which should be processed by the kernel.

Round Robin distribution results in a perfect load balance, but when the packets are further processed it might be useful to capture comprehensive flows. If round robin is used like this, the packets are captured with no relation to each other. Using the identification field of the IP header as an input for a hash function might also result in a balanced load, but to find a function that allocates all packets of a flow to one node and balances the load on all nodes might be very hard to find.

Altogether DiCAP is a well designed architecture, where the authors spent time on thinking about good solutions for scalability, fault tolerance and performance. But to be useful for network operators who want to monitor packets in order to analyze them, another distribution method has to be found.

3. Comparison of different architectures

In order to get an overview of the different forms of distributed architectures, recent proposals of other scientists are presented and compared in the way they handle the upcoming problems of such a solution. It has to be said that comparison is somehow difficult because technology made such big steps over the years. Some implementations handle 100 Mbit network links, others 10 Gbit links. Also the different architectures are often designed for different purposes. The next sections will compare the design, the way packets are distributed and how the capture nodes work.

3.1 Architecture

All studied architectures have in common that they are designed for high speed network links. Normally the packets are copied by a mirroring device or an Ethernet switch. An anomaly of DiCAP is the lack of an active distribution device. Most other designs include an active distributer. For this task a router or an Ethernet switch is neccessary. The mirroring device of DiCAP has a similar task, but it does not decide which capture node gets which packet, the traffic is just forwarded to all capture nodes. To spread the selection process to the single capturing nodes which execute the given rules, like DiCAP does, might help to avoid a bottleneck at the distribution device.

DiCAP uses a coordinator node to organize the capture cluster, other approaches mainly miss a coordinating device. In [6] a manager device is used to advertise definitions for analyzing packets, collect reports and add or remove capturing devices. Flexibly adding and removing capture nodes, like it is possible in DiCAP is not possible in other architectures like [12]. Since they mostly use capturing devices explicitly defined for a special scope.

With the exception of DiCAP, almost all examined approaches combine the capture device and the analyzing device. This leads to a higher processor load of the individual nodes but avoids the need of transferring captured data to an analyzing device and an eventually required distribution process. This design decision is a question of processing power and the number of analyzing nodes, thus a question of traffic per node. An external storage center as proposed in [8] will be necessary when capturing traffic on high speed links for longer periods of time.

3.1.1 Dispatch Method

The critical concern of the distributed designs is the question of how to dispatch the traffic. There are some major principles the method should fulfill:

- In order to minimize packet loss and regarding the processing power limitations, load balancing has to be achieved.
- To draw comprehensive conclusions of the captured packets, the dispatch process has to consider the logical memberships of the packets, such as flows, source or protocols used.

As already presented above, one method is to use a round robin distribution method that simply spreads the traffic into even parts by allocating every n-th node the mod n-th packet. It is a cheap process and may be sufficient for the first criteria mentioned. But it is not suitable for the second one. A solution might be to add an analyzing device that creates a kind of reassembly list which

enables reconstruction of sessions for example [12]. Since every packet has to be identified, a big overhead would be the result.

Most other approaches categorize the packets by one of their attributes and not by the order they are sent. Splitting the traffic by their destination port is an often realized suggestion, since it fulfills the second criteria. When packets are dispatched like this, it is for example possible to reconstruct TCP sessions or to analyze a flow path. Load balancing is hardly given. Since there are far more HTTP packets than FTP packets transmitted there is likely to be an imbalance between the workload of the different capturing devices. A solution is to allocate different numbers of ports to each node, so that the estimated load is equal for every capturing node. Even one port could be divided between two nodes to capture. The authors of [8] developed a system that sorts the packets by their IP source addresses and allocates them to different nodes by using a greedy algorithm. Estimated traffic for each IP source address segment is appraised and every node gets a similar amount of estimated traffic allocated. If the traffic is monitored longer the load balance is almost existent. A promising way to reduce load per node is to use different stages of processing, so two packet attributes could be used to dispatch the traffic. First the packets could be split by their port destination and in a second layer packets could be dispatched by their source IP address (see Figure 4). This method ensures the second criteria, since a flow is identified by its IP source\destination address, the source\destination port and a layer four protocol. So packets within the same flow will be captured at the same node. Also great potential for load balancing exists because the traffic can be divided and load balanced on two layers. The authors of [5] are using this method to analyze packets.

Another suggestion is to assign an identifier to every flow and every node a field of responsibility which flows to capture [10]. This way the captured packets could be distributed evenly by using a round robin algorithm and the packets of one flow could be all captured at the same node.

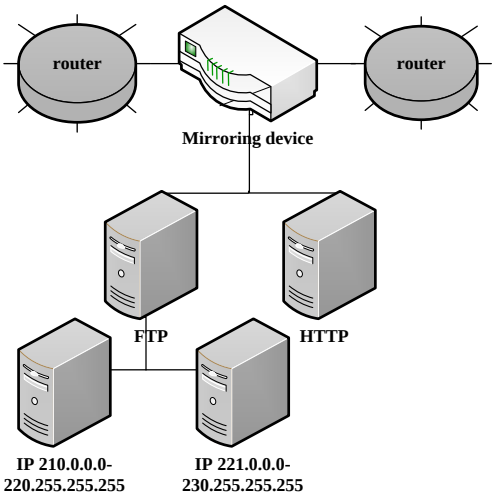


Figure 4: two layer distribution

A problem with attribute dependent dispatch methods is the susceptibility to denial of service attacks. For example, if the network is flooded with packets using just one port. In contrast to round robin methods the attribute filtering is inflexible towards increasing traffic. Round robin methods, like the one used in DiCAP, distribute additional packets always evenly between the different nodes. Another drawback of the attribute filtering methods is the low expandability compared with round robin. New capturing nodes cannot be just added, since every new node has to be assigned to a special attribute. Only the existing nodes which use the same attribute as the new node will be relieved. In round robin every capture node is relieved, when a new node is added.

3.1.2 Capturing Device

In order to achieve scalability, the capture devices are normally standard PCs providing several interfaces. In some proposals the capture nodes also serve as an analyzing device, others forward the captured data to an external analyzer. The module to decide whether to capture a packet or not should be implemented as low as possible in the capturing device [9] in order to avoid wasting processing power on useless packets. Most drafts are based on Linux kernels and use a software tool like libpcap[7] to capture packets.

3.2 Performance

As mentioned before a comparison of the different proposals is hard because of different hardware and software used and different composition and amount of traffic monitored. No independent evaluation was possible, so all numbers are based on the data of the different authors. This paper mainly presents the evaluation results of several architectures presented.

The authors of [8] show that their implementation of a distributed architecture, using a round robin method to dispatch the traffic, experiences no packet loss using four capture nodes, where a single centralized capture device loses 90 percent of the packets. An IDS architecture [12] using seven nodes, a dispatch method dependent on the destination ports of the packets and an optimized number of Snort [13] rules for each capture/analyzing node. This method reduces the number of snort rules on each node, by checking only the Snort rules which are relevant for each port. This method performs up to ten times faster than a single centralized device. The authors of [6] show that a division of flow comparison patterns increases the packet capture rate. Evaluation of DiCAP has shown that libpcap on one node is not able to capture every packet. By using the distribution mode of DiCAP and round robin selection, the packet capture rate increases by the number of nodes. Figure 5 shows the packet capture rates for different numbers of nodes at different rates of packets per second. At some rates four nodes capture ten times more packets than one node.

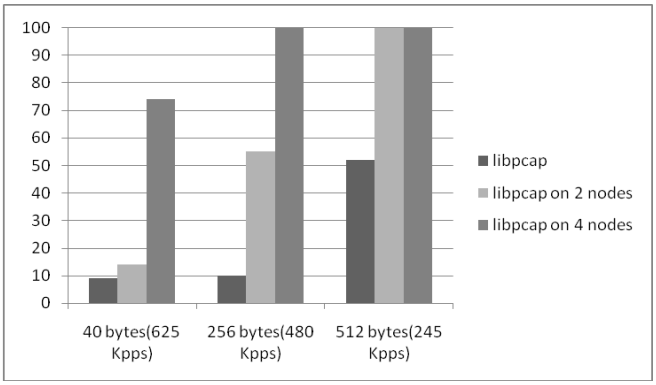


Figure 5: Packet capture rate of libpcap[9]

The results demonstrate that packet loss rates or time for analyzing packets is decreased by taking advantage of multiple capture devices and splitting up the rules to analyze packets. Distributed architectures always outperform centralized ones, when the same hardware is used. It has to be considered that distributed architectures consist of several capture or analyzing devices. Several devices provide more processing power than a single device. The weakness of many evaluations studied is that they only compare their architectures to single devices. A real boost in performance is only existent if the architecture does not only distribute the workload, but also optimize the processing of the workload. An example for such architecture is presented in [12].

4. CONCLUSION

The reviews of the different proposals made on distributed architectures designed for high speed networking monitoring and analyzing have shown that they outperform centralized architectures using comparable hardware. They are scalable, flexible, use standard hardware and open source software. Defects in distribution methods cause an inaccurate analysis or an unequal load balance, therefore high performance hardware is currently preferred by potential customers. Yet the fast development of the web and bandwidth causes a growing gap between network speed and computing speed. I think this gap can only be closed by introducing distributed architectures, if needed combined with high performance hardware.

Future work has to concentrate on improving dispatch methods, since this is a bottleneck. Recent approaches are not sufficient in meeting the criteria for network monitoring and analyzing. New ideas have to be tested, like this peer to peer method [10], to fully utilize given resources.

5. REFERENCES

[1] Brauckhoff, D., Tellenbach, B., Wagner, A., May, M., Lakhina, A., 2006, Impact of Packet Sampling on Anomaly Detection Metrics, 6th ACM SIGCOMM Conference on Internet Measurement, Rio de Janeiro, Brazil, October 25-17, 2006, pp 159-164.

[2] Cisco Systems inc., 2009, Cisco Visual Networking Index Forecast, White Paper

[3] DaSAHIT Project Homepage, http://www.csg.uzh.ch/research/dasahit, March 2010

[4] Endance Homepage DAG Cards, http://www.endace.com/endace-dag-high-speed-packet-capture-cards.html, March 2010

[5] Han, S., Kim, S., Ju, H. T., Hong, J. W. K., 2002, The Architecture of NG-MON: A Passive Network Monitoring System for High-Speed IP Networks, 13th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, Montreal, Canada, October21-23, 2002, pp 16-27.

[6] Kitatsuji, Y., Yamazaki, K., 2004, A Distributed Real-time Tool for IP-flow Measurement, International Symposium on Applications and the Internet, Tokyo, Japan, January 26-30, 2004, pp 91-98.

[7] Libpcap/tcpdump homepage, http://www.tcpdump.org/, march 2010.

[8] Mao, Y., Chen, K., Wang, D., Zheng, W, 2001, Cluster-based Online Monitoring System of Web Traffic, 3rd International Workshop on Web Information and Data Management, Atlanta, Georgia, U.S.A., November 9-10, 2001, pp. 47-53.

[9] Morariu, C., Stiller, B., 2008, DiCAP: Distributed Packet Capturing Architecture for High-Speed Network Links.

[10] Morariu, C., Stiller, B., 2007, A Distributed Architecture for IP Traffic Analysis, Lecture Notes in Computer Science.

[11] Pf_ring homepage, http://www.ntop.org/PF_RING.html. march 2010.

[12] Sallay, H., AlShalfan, K., Fredj, O. B., 2009, A scalable distributed IDS Architecture for High speed Networks, IJCSNS VOL.9 No.8, August 2009.

[13] Snort homepage, http://www.snort.org/, march 2010.

[14] Szeby, T., 2005, Statistical Sampling for Non-Intrusive Measurements in IP Networks, Ph. D. Thesis, Technische Universität Berlin, Universitätsbibliothek ,Fakultät IV Elektrotechnik und Informatik.

Purpose and security analysis of RASTER

Oliver Gasser
Advisor: Christian Grothoff
Seminar Future Internet SS2010
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: gasser@in.tum.de

ABSTRACT

In this paper we survey the purpose of the RASTER routing protocol and study its input on a P2P network's security. Routing in P2P-systems using adaptive distributed hash tables (DHT) poses different challenges than routing in traditional deterministic DHT-based P2P systems. Routing algorithms designed for deterministic DHTs do not always find those shortest paths in adaptive networks. The routing protocol RASTER tries to solve this problem. It minimizes the storing and forwarding overhead and has a good routing performance. We study RASTER(a)'s performance and examine the impact of faulty or malicious nodes on a P2P system which uses the RASTER routing protocol and discover that such a system is more vulnerable to routing manipulation attacks or erroneous nodes.

Keywords

Distributed hash table, Security, Adaptive DHT, Randomized DHT, Routing

1. INTRODUCTION

1.1 Terminology

Participants in a Peer-to-Peer (P2P) network are called nodes. A hash table matches a key k to its value v. A distributed hash table (DHT) is a hash table distributed over multiple nodes. A routing protocol for DHT tries to retrieve the value v for a certain key k as fast and efficiently as possible. If a certain v is not found at the node itself the node has to make a decision to which neighbor it forwards the query.

In a traditional *deterministic* DHT [6, 7, 9] each node selects its neighbors deterministically. This means, that a node has no influence over the nodes which will be its neighbors. Each node selects a certain fixed number of neighbors, the degree k of the overlay structure. Neighbors are selected based on their position in the overlay's hash space. A drawback of this procedure is that it does not consider the heterogeneity of nodes, i.e. the differences in nodes' Internet connectivity, service capacity, lifetime in the system or willingness to perform a certain task.

Adaptive DHT P2P systems overcome this shortcomings by giving nodes some flexibility to select their neighbors. Specifically a node chooses its neighbors based on characteristics such as their Internet connectivity, service capacity, lifetime in the system and service willingness. The nodes are, however, not chosen totally at random as the term *randomized* DHT may suggest in Wang et al.'s paper [11]. Total randomness in neighbor selection would lead to some problems which will be discussed in Section 3.1. Each node always has some near neighbors in order to ensure that a request can be routed towards its destination without looping back and forth. It has been shown [4] that picking neighbors with certain randomness leads to overall better performance specifically in terms of path latency, resilience to churn and local convergence.

1.2 Routing in adaptive DHTs

Unfortunately greedy routing, the dominant routing strategy in deterministic DHTs, falls short of finding the shortest overlay paths in adaptive DHTs. This strategy forwards queries to neighbors who are thought to be closest to the destination node, but do not respect the DHT's nondeterministic overlay structure. The Neighbors of Neighbors (NoN) strategy proposed by Manku et al. [5] performs better than greedy routing but is still far from optimal. In order to get better routing performance, NoN routing could be generalized, i.e. nodes also save 3rd, 4th etc. degree neighbors in their routing tables. However, this creates a huge space and communication overhead.

RASTER [11], a routing protocol for adaptive P2P networks, addresses this problem. It solves the overhead problem caused by having an exponentially increasing number of neighbors in routing tables, by aggregating routing states and encoding routing tables as *bitmaps*. It maintains routing information within a radius of a. It outperforms NoN and is more resilient to churn (nodes joining and leaving the network). RASTER guarantees the discovery of shortest-path overlay routes if the destination d is within a hops, but not otherwise.

In this paper we are describing RASTER in more detail. We then present measurement data investigating the claim of RASTER(a)'s better performance and survey the performance impact of changing parameters. Additionally we are also examining the impact of more information on the neighboring topology on a P2P network's security. If a malicious node joins a network which uses the RASTER algorithm, it has a greater influence factor than by joining a network relying solely on greedy routing.

The rest of this paper is structured as follows. In Section 2 we will examine Chord [9] as an example of a routing algorithm for deterministic DHTs and then in Section 3 we will

take a look at general routing strategies and see why greedy routing encounters problems in adaptive DHTs. Thereafter, as a possible solution for these problems, the bitmap concept and the RASTER routing algorithm [11] are introduced in Section 4. In Section 5 we do a performance comparison of RASTER's two different forwarding decision implementations. In Section 6 we discuss the impact of faulty or malicious nodes and we conclude this paper in Section 7 with the result of our inquiries.

2. DETERMINISTIC DHT ROUTING

This section will describe Chord [9], a typical, simple, deterministic DHT routing algorithm.

In Chord, a hash function assigns each node an m-bit ID by hashing the node's IP address. IDs are ordered in an *identifier circle* modulo 2^m. The key k is assigned to a node as follows: (1) If a node n exists with the same ID as the key k, then k is assigned to n, or (2) if there is no such node, then k is assigned to the successor node or $successor(k)$ which is the first actual node on the circle following k's ID.

For key location purposes each node n maintains a *finger table*, a routing table with m (number of bits of the hash space) entries. Each entry in this table consists of a Chord ID and an IP address. The i^{th} entry in the table points to the node $s = successor(n+2^{i-1})$ for $1 \leq i \leq m$. This is the first node which succeeds n by at least 2^{i-1}. Note that these neighbors in the finger table are selected deterministically.

In our example the nodes' IDs have a length of 6 bit ($m = 6$) and there are 10 nodes in the system. Figure 1 illustrates the finger table entries of node 48.

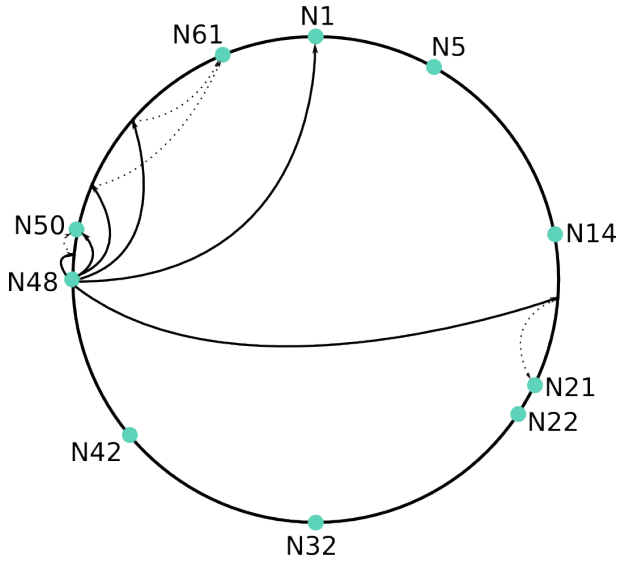


Figure 1: Chord neighbors of node with ID 48.

If a node n gets a request to retrieve the value v for the corresponding key k it first checks if itself is responsible for this key, i.e. $k \in (n, successor(n))$. If that is the case v is either stored at n and returned right away or can not be found in the DHT. Otherwise n forwards the request to the neighbor with the largest ID m smaller or equal to k. More formal:

$$\max_{m \in FT} \{m \leq k\}$$

where FT is the set of finger table IDs. This algorithm is of complexity $\mathcal{O}(\log n)$ as each node has $\log n$ neighbors and after at most $\mathcal{O}(\log n)$ hops the node responsible for the value v has been found. Each forwarding decision at least halves the distance to the destination.

3. ROUTING STRATEGIES IN DHTS

3.1 Greedy routing

Greedy routing is a strategy which bases its forwarding decision as follows: A node n forwards a request to that neighbor W^*, which is closest to the destination d. In a more formal way it selects W^* of its neighbors $W_1, W_2, \ldots, W_k$ for which $|d - W^*|$ is minimal. Assuming that the out degree of each node is $\mathcal{O}(\log n)$, the storage and forwarding overhead for each node in a deterministic DHT is $\mathcal{O}(\log n)$, the average path length between two arbitrary nodes is $\mathcal{O}(\log n)$.

As shown in Section 2, nodes in deterministic DHT networks select their neighbors based on a specific algorithm. For demonstration purposes we assume that the nodes are aligned like a matrix in the hash space and two nodes are neighbors, if they are side by side in a matrix' row or column. In an environment with deterministic neighbor selection, the greedy route always leads to shortest overlay paths (see Figure 2 (a)). Algorithms that are based on the greedy routing strategy do not find shortest paths in adaptive DHTs, i.e. with a somewhat irregular overlay structure. By forwarding the message to a neighbor they come closer to the destination. However, it is not guaranteed that this really is the route with the fewest hops. It is possible that the neighbor, to whom the request has been forwarded to, has neighbors which come only a little closer to the destination node d (see Figure 2 (b)).[1] The origin of the problem is that although you make a locally optimal choice, i.e. you make the biggest step towards the destination, you can't assume the same about it globally as in adaptive networks you don't have sufficient knowledge about the structure of the node's neighbors, the neighbors' neighbors etc.

Greedy routing (e.g. Chord), however, is not optimal for adaptive networks, i.e. where each node chooses its neighbors not deterministically but according to the neighbor's latency, service capacity, lifetime in the DHT etc. To find a better solution than greedy we need to survey the different requirements for routing in deterministic and adaptive DHTs.

3.2 NoN routing

An approach similar to greedy but specifically designed for adaptive DHTs is *Neighbor of Neighbor routing* (NoN) proposed by Manku et al. in [5]. In this algorithm each node not only saves information about its neighbors but also of its neighbors' neighbors (also called neighbors of 2^{nd} degree). When making a decision to which node to forward a request, the node forwards it to neighbor W^*, which has itself

[1] If the nodes chose neighbors at random as the term randomized DHT in [11] suggests, it would even be possible that there is no neighbor which gets locally closer to d, resulting in the wrong peer being identified as the target.

a neighbor closest to the destination node d. I.e. n identifies W^*, which can be of n's 1^{st} and 2^{nd} degree neighbors, and forwards the request to W^*. The average path length between two arbitrary nodes with NoN routing in a DHT where each node has $\mathcal{O}(\log n)$ neighbors is $\mathcal{O}(\frac{\log n}{\log \log n})$. This performance improvement compared to greedy routing comes with a significant drawback: Each node's routing tables contain $\mathcal{O}(\log^2 n)$ entries and the communication overhead is also greater.

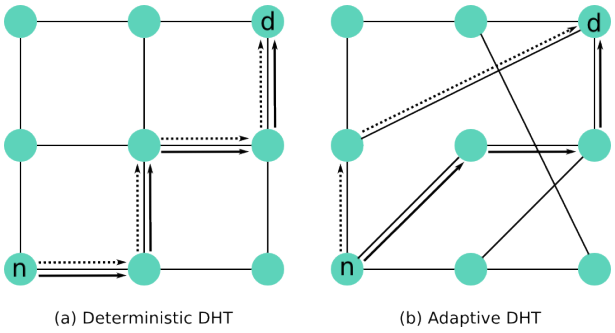


Figure 2: Difference between shortest path route (dashed line) and greedy route (solid line).

The NoN strategy performs slightly better than greedy routing as it can avoid neighbors which are perceived to be close to the destination node d but where then their neighbors are not rapidly progressing towards d. NoN is not the panacea as it only includes neighbors of 1^{st} and 2^{nd} degree. As pointed out in [11] (1) the average number of overlay hops independent of the overlay structure approaches the Moore bound $\lceil \log_k n \rceil$, where n is the number of nodes and k the degree. (2) The performance gap between shortest route and chosen route can be bridged by generalizing NoN routing, i.e. nodes also maintain information about higher degree neighbors. This, however, is not practical as it comes with high communication and storage overhead. The RASTER routing protocol, which will be explained in detail in the next section, addresses this issue.

4. RASTER ROUTING PROTOCOL

This section describes the definition and reasoning behind using bitmaps instead of traditional routing tables, construction of and operations on such bitmaps and the RASTER decision procedure in detail.

4.1 Bitmaps

A *bitmap* is an encoded representation of a set of nodes in a DHT network. It is important to note that the size of the bitmap is independent of the number of nodes in the DHT.

DEFINITION 1. *$B_s^{a,r}$ is a bitmap of resolution r, from an arbitrary node s and covering a radius a. It is an r-bit binary string representing the overlay positions that can be reached from node s within a hops. The system's hash space is to be partitioned evenly into r bins of the same size. If the i^{th} bit $(0 \leq i \leq r-1)$ in $B_s^{a,r}$ is set to 1, at least one node in bin i is reachable within a hops from node s.*

$B_s^{1,r}$ therefore represents the overlay neighbors of node s, $B_s^{2,r}$ neighbors of 2^{nd} degree and so on. See Figure 3 and

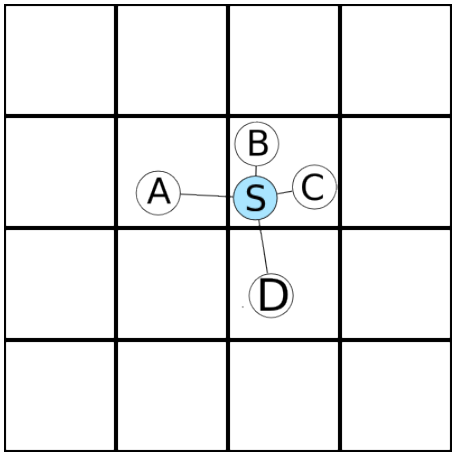


Figure 3: Constructing $B_S^{1,16}$ = 0000 0110 0010 0000.

Figure 4 on how to construct bitmaps. Contrary to traditional routing tables these bitmaps hide routing details, which are not relevant for forwarding decisions. Bitmaps don't contain entries for every single node but combine them into bins. Furthermore a node only retains IP addresses for its direct neighbors and not for all routing table entries. This makes bitmaps much smaller than routing tables.

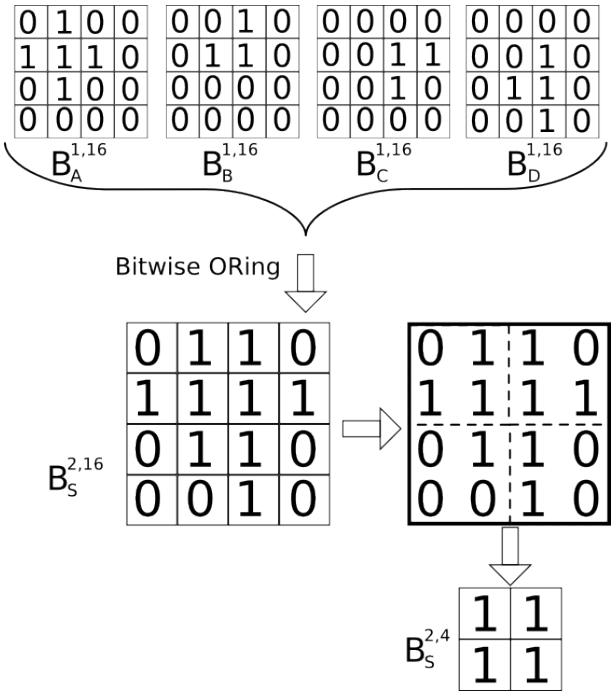


Figure 4: Constructing $B_S^{2,16}$ = 0110 1111 0110 0010 and $B_S^{2,4}$ = 1111

4.2 RASTER routing

For RASTER to be able to make a forwarding decision, each node needs to store certain bitmaps. RASTER(a) defines the algorithm, where each node s maintains bitmaps about its neighboring topology within a radius of a: Each node s keeps $B_s^{1,r_{max}}, B_s^{2,r_{max}}, \ldots, B_s^{a,r_{max}}$ and addition-

ally $B_k^{1,r_{max}}, \ldots, B_k^{a-1,r_{max}}$ for all neighbors k, where r_{max} is the maximum resolution.

The storage amount for each node is therefore $a \cdot r_{max}$ bits for its own bitmaps plus $k \cdot (a-1) \cdot r_{max}$ for its neighbors, where k is the out-degree, i.e. number of neighbors. Assuming that each node has a logarithmic number of neighbors ($k \in \mathcal{O}(\log n)$), the total storage cost for one node is $(a + \log n \cdot (a-1)) \cdot r_{max}$ bits.

The general approach of the routing decision procedure is as follows. When a node s receives a query for a key k, it first identifies the corresponding bin in the system's hash space, i.e. the bit in the bitmap, b_0. The node then forwards the query to its neighbor k^*, which (1) can reach the closest bin to b_0 and (2) following the minimum number of hops.

4.2.1 Straightforward implementation

Upon receiving a query node s searches $B_s^{1,r_{max}}, B_s^{2,r_{max}}, \ldots, B_s^{a,r_{max}}$ to find the closest bit to b_0 set to 1. If is b_0 found in more than one bitmap, choose the one with the smallest radius H. RASTER stops looking when the bit representing the destination bin is set. This means that there is a neighbor which can reach the closest bin in $H-1$ hops. To find this neighbor we look through $B_k^{H-1,r_{max}}$ for all neighbors k until one of them, k^*, has set the bit b_0 to 1. This is the neighbor to which the query will be forwarded to. Wang et al. [11] claim, that this approach had a significant drawback: Searching through all the bitmaps for the bit closest to b_0 is expensive. The *real* RASTER(a) implementation solves this problem by maintaining and searching through different resolution bitmaps. We will survey this claim in Section 5.

4.2.2 RASTER(a) implementation

The idea of this implementation compared to the straightforward one is to keep multiple resolutions for each bitmap. First you determine where the queried key resides for all maintained resolutions. Then, instead of searching all bitmaps with resolution r_{max}, different resolution bitmaps are searched for the bit closest to b_0. See Figure 5 for a visual representation.

4.2.3 Processing overhead

In an adaptive DHT network with n nodes and an out-degree of $k \in \mathcal{O}(\log n)$ which implements RASTER(a) maintaining l resolutions, has the following overheads [11]: To make the *forwarding decision* $\mathcal{O}(\log n)$ bitwise operations are necessary. The cost of *exchanging bitmaps* is $\Theta(a \cdot r_{max} \cdot \log n)$ bits and the cost of storing this information is $\Theta(a \cdot l \cdot r_{max} \cdot \log n)$ bits.

5. ROUTING DECISION PERFORMANCE COMPARISON

In this section we will compare the decision process' performance of the straightforward implementation and the more elaborate RASTER(a). These benchmarks written in Java were conducted on a regular consumer laptop.[2]

[2]Intel Core2Duo T7500 @ 2.21 GHz, 2 GiB RAM, Ubuntu 9.10, source code found at [3]

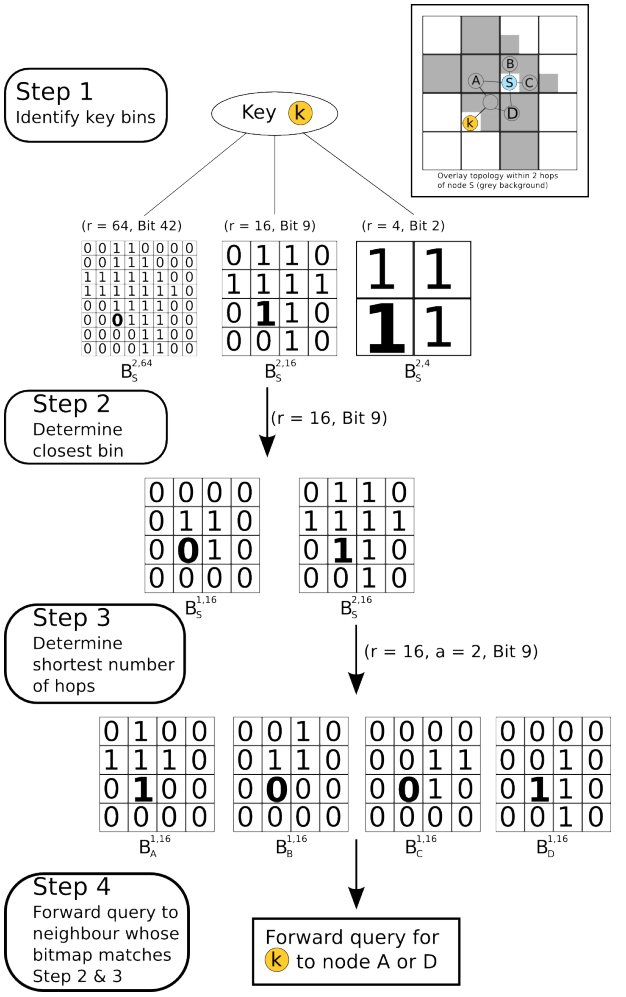


Figure 5: Node S' forwarding decision following RASTER(2): (1) Identify the destination bin's bit for every resolution (bold bits). (2) Find the largest resolution where this bit is set ($r = 16$). (3) Find smallest radius for which this bit is set ($a = 2$). (4) Find neighbor who can reach the destination bin in $a - 1 = 1$ hops.

5.1 Benchmarks

In a DHT with 80,000,000 nodes, an average out-degree of 50, a maximum resolution of 100 and a radius of 3, the straightforward implementation (SFWD) takes on average 27 milliseconds to perform 1 million routing decisions. The "faster" RASTER(3) takes 39 ms (with RASTER(3) using 3 additional scaled down resolutions by halving). In the following paragraphs we will look at how both algorithms react to parameter changes. The fixed parameters for the following benchmarks have the values mentioned at the beginning of this paragraph.

If we add more nodes to the DHT, resulting in more average overlay neighbors, both SFWD's and RASTER(a)'s time slowly increases. SFWD's time to calculate forwarding decisions rises faster and surpasses RASTER(a) at 83 neighbors (see Figure 6).

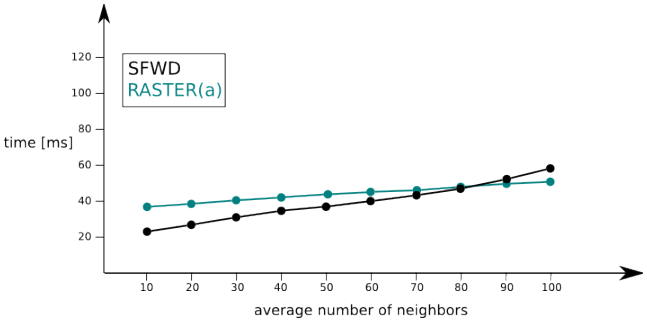


Figure 6: Performance impact of changing the number of neighbors.

By decreasing the bitmaps' resolution, SFWD's time to make a lookup decision decreases greatly, increasing the resolution increments this time. SFWD is thus sensitive to a resolution change. RASTER(a) is much less sensitive to this change, it begins to outperform SFWD at a resolution of just over 300 bits (see Figure 7).

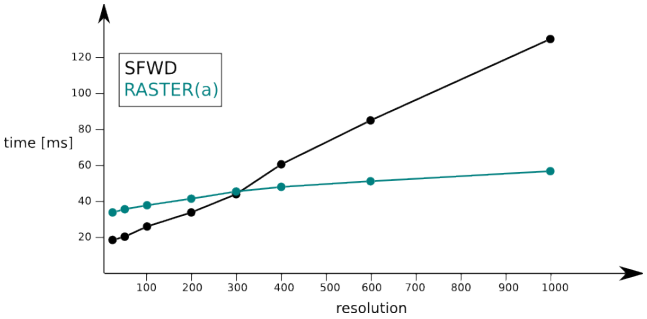


Figure 7: Performance impact of changing the resolution.

Altering the radius also results in SFWD's performance scaling slightly worse than RASTER(a)'s. See Figure 8 for a visual representation.

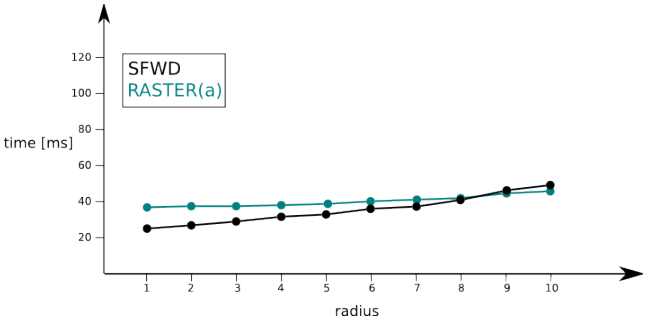


Figure 8: Performance impact of changing the radius.

5.2 Results

The result of these inquiries are that the RASTER(a) implementation is not generally faster than the straightforward one, although the former is of the order of $\mathcal{O}(\log r)$ and the latter is in $\mathcal{O}(r)$ and the performance of both is impacted by parameter choices.

The more interesting observation, however, is that RASTER(a) is a rather premature optimization. SFWD takes 27 ms to make 1 million forwarding decisions, which is approximately 40 million packets per second. Assuming a packet size of 1 kB this corresponds to 40 GB/s. The forwarding decision algorithm is thus — for moderately large resolutions r — not the bottleneck of systems implementing the RASTER protocol. For higher resolutions, the security issues raised in the following section are likely to be a bigger concern than performance.

6. RASTER SECURITY

One of RASTER's concerns, which is ignored in [11], is the aspect of security. In this section we will survey the impact of faulty or malicious nodes on RASTER.

6.1 Assumptions

Our inquiries are based upon these basic assumptions: (1) A malicious node is able to join the DHT network. In a decentralized DHT it is due to the absence of a central authority inherently difficult to prevent a malicious node from joining the network. (2) The malicious node is able to manipulate data (specifically change a bitmap), which resides on its system and by forwarding it to another node also modifying this node's bitmap data. (3) The malicious node tries not to be detected by other nodes. The malicious node therefore needs to adhere to the protocol's specification insofar as other nodes wouldn't get suspicious and ignore the malicious node.

6.2 Vulnerabilities

To understand RASTER's vulnerabilities we need to focus on its characteristics. One of them is that each node keeps special routing tables, the so called bitmaps. These bitmaps do not only contain information about direct neighbors but of neighbors of degree a. This is the main weakness of RASTER.

6.2.1 More neighbors

In an attack a malicious node could simply manipulate its bitmaps, so that it appears that it is able to reach a lot *more* neighbors within a radius a than it actually can. As a consequence more requests than usual will be forwarded to this node. The malicious node can then perform a series of attacks such as dropping the request, delaying it, sending a wrong response or compromise user's privacy by logging all activity. Possible attacks are the *Sybil* attack [2], where an attacker needs less nodes joining the DHT to get a certain control over it as nodes using bitmaps have a greater manipulation capacity. The same applies for the *Eclipse* attack [8]. Possible solutions to this attacks are listed in [10] and are outside the scope of this paper.

To understand how many additional requests a node receives, we need to discuss how the parameters a and r are best chosen. To have as much information entropy, i.e. encoded information, as possible in a bitmap half of its bits should be set to 1 and the other half to 0. Then we have to define a maximum influence factor x, that is the number of nodes one malicious node can influence. The radius is then chosen as

$$a = \lfloor \log_k 2 \cdot x \rfloor$$

where k is the average number of neighbors. The number of nodes a malicious node can influence is therefore bounded by $\frac{k^a}{2}$. As the resolution is chosen such that half of the bits are set to 1, an attacker can only set the other half to 1 as well, which corresponds to 50% of the nodes an attacker can possibly reach at the maximum radius.

The resolution is than chosen as

$$r = 2 \cdot k^a$$

to ensure that approximately half of the bits are set to 1.

If you have sufficient knowledge about the nodes participating in the DHT you can choose a threshold of maximum 1s set in order to detect and subsequently ignore malicious nodes. A malicious node could then always set as many bits to be just under the threshold and will thus not be detected. Thresholds work especially good if all nodes have very similar service capacities as the threshold is then just a fraction above the ideal 50% of 1s. A malicious node could still undercut this threshold but it can only influence fewer additional nodes than in a DHT with a diverse service spectrum.

6.2.2 Fewer neighbors

The contrary, i.e. manipulating its own bitmap so that it appears that the malicious node can reach *fewer* nodes within a radius a, is also a problem. By pretending that a node cannot reach any or very few neighbors, it avoids being selected to forward queries. A node therefore has to forward almost no routing decisions for other neighbors but its requests are granted without any objection. This phenomenon is called *free riding* or *leeching* and is discussed in detail in *Free Riding on Gnutella* [1]. This not only leads to inequality among nodes but it also compromises a DHT's security by diminishing the availability.

6.3 RASTER vs. greedy security

The difference between RASTER's bitmaps and greedy's normal routing tables is that in the former one node has a much larger manipulation influence radius than the latter. Specifically one node in a DHT with greedy routing can only influence its direct neighbors by manipulating its routing table. Thus the number of possibly manipulated nodes in a greedy routed DHT with n participants is $\mathcal{O}(\log n)$. Contrary to that a node in a DHT implementing RASTER's bitmaps, depending on the radius a, is theoretically able to manipulate a lot more nodes: $\mathcal{O}(\log^a n)$. The number of nodes a malicious node can manipulate, grows *exponentially* with the radius a.

To make this contrast more obvious let us look at the following example. Suppose we have a DHT with $n = 1000$ nodes and an out degree of $k = 5$. If using greedy routing, one node can manipulate just 5 nodes', i.e. its direct neighbors, routing decisions which corresponds to 0.5% of all the nodes. However, if the DHT uses RASTER(4), one malicious node could potentially influence the forwarding behavior of $5^4 = 625$ nodes which corresponds to 62.5% of all nodes in the DHT. One single node is able to regulate more than half of the DHT.

The security impact can be bound by restricting a as suggested in Section 6.2.1 or by comparing the number of set bits to network topology statistics.

7. CONCLUSION

We discussed the benefits of using adaptive DHT networks in contrast to deterministic DHTs. It is however suboptimal to use algorithms, which were designed for deterministic DHTs, for adaptive networks. As an example we explained the Chord algorithm and we described that the RASTER algorithm, which was specifically devised for adaptive DHTs, has many advantages over greedy based strategies. We then compared the performance of both RASTER implementations and concluded that the choice of forwarding algorithm does not really matter. Regarding security, algorithms, in which one node has a lot of information about its neighboring topology and exchanges this information with them like RASTER, are inherently more susceptible to malicious or faulty nodes. We have seen, that one malicious node in RASTER can manipulate up to $\mathcal{O}(\log^a n)$ additional neighbors' routing decisions. By selecting the parameters carefully it can be reduced to $\frac{k^a}{2}$.

8. REFERENCES

[1] E. Adar and B.A. Huberman: *Free riding on gnutella*, In First Monday, pages 2–13, Citeseer, 2000

[2] J. Douceur: *The sybil attack*, In Peer-to-Peer Systems, pages 251–260, Springer, 2002

[3] O. Gasser: *Source code used to compare performance of straightforward implementation and RASTER(a)*, `http://projects.net.in.tum.de/downloads/raster.tgz`

[4] J. Kleinberg: *The small-world phenomenon: an algorithm perspective*, In Proceedings of the thirty-second annual ACM symposium on Theory of computing, pages 163–170, ACM New York, NY, USA, 2000

[5] G.S. Manku and M. Naor and U. Wieder: *Know thy neighbor's neighbor: the power of lookahead in randomized P2P networks*, In Proceedings of the thirty-sixth annual ACM symposium on Theory of computing, pages 54–63, ACM New York, NY, USA, 2004

[6] P. Maymounkov and D. Mazieres: *Kademlia: A peer-to-peer information system based on the xor metric*, In Peer-to-Peer Systems, pages 53–65, Springer, 2002

[7] A. Rowstron and P. Druschel: *Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems*, In IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), pages 329–350, Citeseer, 2001

[8] E. Sit and R. Morris: *Security considerations for peer-to-peer distributed hash tables*, In Peer-to-Peer Systems, pages 261–269, Springer

[9] I. Stoica and R. Morris and D. Karger and M.F. Kaashoek and H. Balakrishnan: *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, pages 160, ACM, 2001

[10] G. Urdaneta and G. Pierre and M. van Steen: *A Survey of DHT Security Techniques*, Citeseer, 2008

[11] C.C. Wang and K. Harfoush: *RASTER: a light-weight routing protocol to discover shortest overlay routes in randomized DHT systems*, In Proceedings of the 12th International Conference on Parallel and Distributed Systems, pages 553–560, 2006

DDoS-Angriffs- und Verteidigungsstrategien

Alexander Wittmann
Betreuer: Marc Fouquet
Seminar Innovative Internet Technologien und Mobilkommunikation SS2010
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: alexander.wittmann@in.tum.de

KURZFASSUNG

Distributed Denial-of-Service (DDoS) Angriffe stellen eine aktuelle Bedrohung für Netzwerkumgebungen dar. Die Vielzahl und Vielfalt sowohl der Angriffs- als auch der Verteidigungsansätze ist beachtlich.

Dieses Dokument präsentiert und erläutert die wichtigsten und bedeutendsten Möglichkeiten zum Angriff und zur Verteidigung, und bietet damit Neulingen die Möglichkeit zum besseren Verständnis des Problems und der aktuellen Lösungsansätze.

Der die Angriffsmöglichkeiten betreffende Teil der Ausarbeitung stellt Gemeinsamkeiten heraus und hebt wichtige Merkmale der Strategien hervor. Der die Verteidigungsstrategien betreffende Teil erläutert die Vor- und Nachteile der Lösungsansätze.

Schlüsselworte

DoS, DDoS, Attack Rate, Source Address Validity, Spoofing, Traceback, Pushback, Source Address Filtering, Overlay Filtering, Anomaly Detection, Client Puzzles

1. EINLEITUNG

Dieses Dokument gibt einen allgemeinen Überblick über Angriffe, deren primäres Ziel es ist, den Zugriff auf eine bestimmte Ressource zu verweigern. Im speziellen werden einige Möglichkeiten vorgestellt, wie auf solche Angriffe reagiert werden kann.

Als Denial of Service (kurz DoS, englisch für: Dienstverweigerung oder -ablehnung) wird in der digitalen Datenverarbeitung die Folge einer Überlastung von Infrastruktursystemen bezeichnet (siehe Abbildung 1).

Solch eine Dienstverweigerung kann durch unbeabsichtigte Überlastungen verursacht werden oder durch einen mutwilligen Angriff auf einen Host (Server), einen Rechner oder sonstige Netzkomponenten in einem Datennetz. Dies geschieht in der Regel mit der Absicht, einen oder mehrere bereitgestellte Dienste arbeitsunfähig zu machen.

Erfolgt solch ein Angriff koordiniert von einer größeren Anzahl anderer Systeme aus, so spricht man von Verteilter Dienstblockade oder englisch Distributed Denial of Service (DDoS). Normalerweise werden solche Angriffe nicht per Hand, sondern mit Backdoor-Programmen oder Ähnlichem durchgeführt, die sich von alleine auf anderen Rechnern im Netzwerk verbreiten und dem Angreifer durch solche Botnetze weitere Wirte zum Ausführen seiner Angriffe bringen.[4]

Nicht alle Dienst-Ausfälle, auch diejenigen, die sich aus bösartigen Aktivitäten ergeben, sind zwangsläufig Denial-of-Service-Attacken.

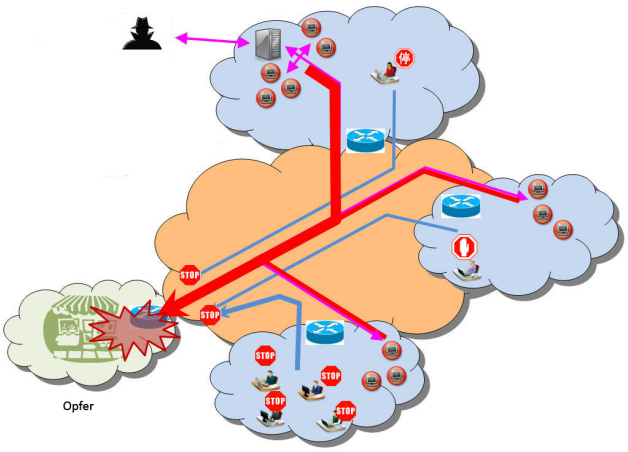


Abbildung 1: DDoS-Angriff

Unrechtmäßige Nutzung von Ressourcen kann auch zu Dienst-Ausfällen führen. Zum Beispiel kann ein Eindringling einen anonymen FTP-Bereich als Ort nutzen um illegale Kopien von kommerzieller Software zu speichern, somit Speicherplatz verbrauchen und die Generierung von Netzwerk-Verkehr verursachen. Derartige Angriffe sollen aber nicht im Rahmen dieses Dokuments behandelt werden.

Denial-of-Service-Angriffe können Rechner und Rechnernetze wesentlich beeinträchtigen. Je nach Art Ihres Unternehmens, kann somit gezielt Ihr Geschäftsbetrieb zum Erliegen gebracht werden. So kam es bereits häufiger vor, dass Anbieter von E-Commerce Diensten erpresst wurden. Sollte einer Lösegeldforderung nicht nachgekommen werden hätte dies eine Blockade der Internetpräsenz durch einen DDoS-Angriff zur Folge. Der entstehende Schaden ist immens.

Einige Denial-of-Service-Angriffe können mit einfachen Mitteln gegen einen leistungsfähigen Host eingesetzt werden. Diese Art von Angriff wird als ein asymmetrischer Angriff bezeichnet. So ist es z.B. einem Angreifer möglich mit ge-

ringem Rechenaufwand und geringer Bandbreite erheblich leistungsfähigere Endgeräte zu beeinträchtigen.

2. ANGRIFFSSTRATEGIEN

Denial-of-Service-Attacken gibt es in einer Vielzahl von Formen. Die Angriffe zielen dabei auf unterschiedliche Dienste ab. Es gibt einige grundlegende Arten von Angriffen. Man kann dabei zwei grundsätzliche Typen unterscheiden:

- Verbrauch knapper, beschränkter oder nicht-erneuerbaren Systemressourcen
- Überflutung des Netzwerks mit Angriffen um legitimen Verbindungsaufbau zu verhindern

2.1 Ausschöpfen von Systemressourcen

Teilnehmer eines Netzwerkes benötigen bestimmte Ressourcen um ihre Dienste anbieten zu können, wie etwa Speicher- und Festplattenplatz, CPU-Zeit, und Datenstrukturen.

Denial-of-Service-Attacken richten sich am häufigsten gegen Netzwerkerreichbarkeit. Das Ziel ist es, Hosts oder Netzwerke an der Kommunikation zu hindern. Ein Beispiel für diese Art von Angriff ist der SYN-Flood-Angriff, der im Folgenden kurz beschrieben wird:

Bei dieser Art von Angriff, baut der Angreifer eine Verbindung zum Rechner des Opfers auf, ohne aber den Verbindungsaufbau zu vollenden. In der Zwischenzeit hat das Opfer eine bestimmte Anzahl an Systemressourcen reserviert, um die bevorstehende vollständige Verbindung vorzubereiten.

Das Ergebnis ist, dass legitime Verbindungen verweigert werden, während die Opfer-Maschine darauf wartet, dass halboffene Verbindungen vollständig aufgebaut werden.

Es ist zu beachten, dass diese Art von Angriff nicht davon abhängt, dass der Angreifer in der Lage ist die vorhandene Netzwerkbandbreite vollständig aufzubrauchen. In diesem Fall verbraucht der Angreifer Ressourcen, welche am zur Verfügung stellen der Netzwerkverbindung beteiligt sind.

Das bedeutet also, dass ein Angreifer diesen Angriff von einer langsamen Verbindung aus gegen einen leistungsfähigen Server in einem sehr schnellen Netzwerk ausführen kann. (Dies ist ein gutes Beispiel für einen asymmetrischen Angriff.)

2.2 Ausschöpfen von Netzwerkbandbreite

Ein Angreifer kann auch in der Lage sein, die komplette zur Verfügung stehende Bandbreite in einem Netzwerk aufzubrauchen, indem er eine große Anzahl von Paketen an das Netzwerk sendet. Typischerweise sind diese Pakete ICMP ECHO-Pakete, unterliegen aber prinzipiell keiner Einschränkung.

Darüber hinaus muss der Angreifer nicht von einer einzigen Maschine aus operieren. Er kann in der Lage sein den gemeinsamen Angriff mehrerer Maschinen zu koordinieren, um einen erheblich stärkeren Effekt zu erzielen. (Dies ist ein gutes Beispiel für einen DDoS-Angriff aus einem Bot-Netz)

3. ANGRIFFSEIGENSCHAFTEN

Bei beiden zuvor genannten Typen lassen sich folgende Angriffseigenschaften unterscheiden:

- Angriffe mit echten oder mit gefälschten Absender IP-Adressen
- Angriffe mit dynamischem Wirkungsgrad, die nicht dazu dienen das Netzwerk unverzüglich lahm zu legen sondern auf Dauer die Bandbreite zu begrenzen ohne erkannt zu werden

3.1 Authentizität der Absenderadresse

Source-IP-Spoofing - also das Fälschen der Absender IP - spielt eine wichtige Rolle bei DDoS-Attacken. Würde es eliminiert werden, könnten viele Arten von DDoS-Attacken durch Ressourcenmanagement-Techniken gelöst werden, was eine gerechte Verteilung der Host- oder Netzwerkressourcen, an jede Quell-IP-Adresse bedeuten würde. Basierend auf der Gültigkeit der Absenderadresse, unterscheidet man zwischen Angriffen mit gefälschten Absenderadressen und gültigen Absenderadressen.

DDoS-Angriffe mit gefälschter IP-Absenderadresse sind die vorherrschende Art von Angriffen, da es ein Vorteil für den Angreifer ist, z.B. um die Zurechenbarkeit zu vermeiden, und um durch die größere Adressvielfalt die Erkennung des Angriffs zu erschweren.

Angreifer können nicht routbare Source-Adressen vortäuschen, von denen einige auf eine Reihe von reservierten Adressen (zB 192.168.0.0/16) zeigen oder Teil eines zugeteilten, aber nicht verwendeten Adressraums eines Netzwerks sind. Angriffspakete, die reservierte Adressen verwenden, können leicht erkannt und verworfen werden, während die Pakete die keine reservierten Adressen benutzen wesentlich schwieriger zu entdecken sind.

Viele Angriffe fälschen zufällige Absenderadressen in den Angriffspaketen, da dies einfach durch die Generierung von zufälligen 32-Bit-Zahlen, mit denen die Pakete versehen werden, erreicht werden kann.

Beim sogenannten Subnetz-Spoofing fälscht ein Angreifer eine zufällige Adresse aus dem zugewiesenen Adressbereich in dem sich die Maschine befindet. Zum Beispiel könnte eine Maschine, die Teil des Netzwerks 131.179.192.0/24 ist, jede beliebige Adresse im Bereich 131.179.192.0 - 131.179.192.255 fälschen. Da Maschinen in einem Subnetz sich das Medium (Ethernet) teilen um die Gateway-Router (First Hop auf dem Weg zur Außenwelt) erreichen zu können, kann Spoofing nur von diesem Router nachgewiesen werden. Es ist unmöglich, die Fälschung irgendwo außerhalb des Gateway Routers zu erkennen.

Angreifer profitieren vom Quell-Adress-Spoofing und sind dazu geneigt, es einzusetzen, wann immer es geht. Es ist jedoch nicht mehr dringend notwendig auf Spoofing zu setzen. Zum einen kann ein großer Teil der Angriffe erkannt werden (siehe später), zum anderen hat das Aufkommen von Bot-Netzen dazu beigetragen, das ein Initiator eines DDoS-Angriffs sich einer Vielzahl von Hosts mit unterschiedlichen Adressen bedienen kann.

So können Angriffe von sehr vielen unterschiedlichen Hosts aus durchgeführt werden. Die hohe Zahl der Angreifer macht es möglich, dass das Paketaufkommen eines einzelnen am Angriff beteiligten Rechners sehr gering sein kann und somit die Wahrscheinlichkeit sinkt, dass dessen singuläre Aktivität als böswillig eingestuft wird. Durch die hohe Zahl der Angreifer kann das Ziel dennoch leicht zum erliegen gebracht werden.[3]

3.2 Dynamik der Angriffsrate

Während des Angriffs sendet jeder der teilnehmenden Angreifer einen Strom von Paketen an das Opfer. Ausgehend vom Mechanismus, welcher die Rate ändert, unterscheiden wir zwischen Angriffen mit konstanter, steigender und schwankender Intensität.

Die meisten bekannten Angriffe gehen mit konstanter Rate vor. Nach dem Beginn des Angriffs erzeugen die Angreifer Pakete in gleichmäßigem Tempo, in der Regel so viele wie ihre Mittel es erlauben. Die plötzliche Flut an Paketen stört die Dienste des Opfers schnell. Dieser Ansatz bietet das beste Kosten-Nutzen-Verhältnis für den Angreifer, da er eine minimale Anzahl von Clients bereitstellen muss, um Schaden zuzufügen. Auf der anderen Seite kann der große, kontinuierliche Verkehrsstrom als Anomalie erkannt werden und erleichtert so die Angriffsentdeckung.

Angriffe mit variabler Paketrate verändern die Intensität des Paketstroms der angreifenden Maschine um die Aufdeckung des Angriffs zu verzögern oder zu vermeiden.

Angriffe, die einen stufenweisen Anstieg der Rate verwenden, führen zu einem langsamen Erschöpfen der Ressourcen des Opfers. Die zur Verfügung gestellten Dienste des Opfers werden über einen langen Zeitraum hinweg immer etwas mehr beeinträchtigt. Somit kommt es zu einer wesentlich verzögerten Erkennung des Angriffs.

Der Schaden dieses Angriffs kann sehr groß sein, da zwar nur ein geringer Teil der Ressourcen beeinträchtigt wird, dies allerdings über einen langen Zeitraum hinweg. Ein Angriff mit hoher Rate und dem Zweck, das Netzwerk zum erliegen zu bringen würde schneller erkannt und beseitigt werden.

Angriffe die mit fluktuierender Rate arbeiten passen die Angriffsrate, ausgehend von dem Verhalten des Opfers oder beruhend auf einem vorprogrammierten Timing an, um der Linderung des Schadens und der Entdeckung zu entgehen.

Bei einem pulsierenden Angriff brechen die Angreifer in wiederkehrenden Abständen die Attacke ab, um sie später fortzusetzen. Wenn dieses Verhalten gleichzeitig von allen Angreifern durchgeführt wird, erfahren die Opfer regelmäßige Service-Unterbrechungen. Wenn die Angreifer jedoch in Gruppen unterteilt sind, die so koordiniert werden, dass eine Gruppe immer aktiv ist, dann erfährt das Opfer kontinuierliche Dienstverweigerung, während es die anhaltende Anomalie gar nicht als solche erkennen kann.[3]

4. VERTEIDIGUNGSSTRATEGIEN

Internet Denial-of-Service (DoS) Attacken fluten begrenzte Ressourcen mit Anfragen und verhindern damit legitimen Nutzern den Zugriff auf diese Ressource. Zu den Zielen gehören die Bandbreite an Netzzugangspunkten und anderen Netzwerk-Engpässen, und auch die Rechen- und Speicher-Ressourcen auf Servern, Clients, Routern und Firewalls. Zum Beispiel werden einige Low-End-Router lahmgelegt, wenn an sie Pings in einer zu hohen Rate gesendet werden, weil die CPU überfordert ist.

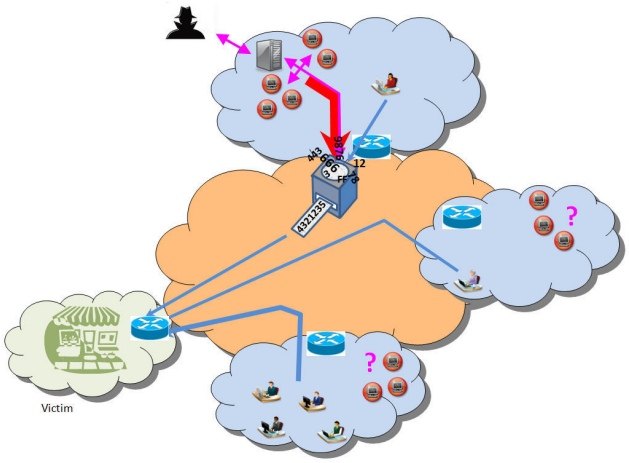


Abbildung 2: Source Address Filtering

Es mag zwar nahe liegen, dieses Problem durch den Einsatz robusterer Software zu lösen. Dennoch ist jedes mit dem Internet verbundene Gerät verwundbar durch einen Flooding-Angriff.

Da DoS-Attacken immer häufiger auftreten wurden viele Wissenschaftler zu Forschungsarbeiten motiviert. Diese Arbeit soll neben gängigsten Angriffsmethoden auch wichtige Erkennungs- und Abwehrmöglichkeiten behandeln.

DoS-Lösungen werden meist anhand ihrer Realisierbarkeit und ihrer Wirksamkeit bei der Abwehr bewertet. Während dies zweifellos richtig ist, soll die Aufmerksamkeit auch auf eine andere Art von Problem auf diesem Gebiet gelenkt werden: die künftigen Folgen für das Internet und seine Anwendungen, falls diese Ansätze zum Einsatz kommen sollten. Ein guter Schutz vor DoS-Lösung muss nicht nur wirksam sein, er muss es auch erlauben neue Netzdienste nahtlos einzuführen. Vor dem Hintergrund aktueller politischer Entwicklungen muss auch beachtet werden, dass die Methoden zur DDoS-Abwehr auch sehr leicht zur Internetzensur eingesetzt werden könnten.

4.1 Reaktive Methoden

4.1.1 Source-Address-Filtering

Einer der frühesten Vorschläge um DoS-Angriffe abzuschwächen war es, Source-Adress-Filtering an allen Netzwerk Einstiegs- und Ausstiegspunkten bereitzustellen (siehe Abbildung 2). Dies würde Angreifer daran hindern beliebige Absenderadressen in ihren Paketen anzugeben und wäre somit nützlich um die Anzahl an Arten der Angriffe zur reduzieren. Ein Source-Filter lässt sich als Filter beschreiben, der Pakete abweist, die einen Punkt im Netzwerk nicht auf legitime Art und Weise erreicht haben können.

Filtering ist nur dann effektiv, wenn es in großem Maße ein-

gesetzt wird. Eine Quelladresse kann den Nachweis für die Urheberschaft nur dann erbringen, wenn jeder Knoten im Netzwerk einen Filter implementiert um die Abgeschlossenheit des Systems sicherzustellen. Obwohl es in den letzten Jahren als best practice galt, gibt es noch viele Lücken bei der Abdeckung mit Adressfiltern. Selbst wenn vollständige Abdeckung gegeben wäre - Fortschritte bei den Angriffsmethoden haben die Relevanz der Filter zur Bedeutungslosigkeit verkommen lassen.

Hinter einem Filter können alle Source-Adressen, die innerhalb eines Netzwerkpräfixes liegen, gefälscht werden. Dies können tausende von Adressen sein. Hinzu kommt, dass automatisierte Botnetz-Tools es leicht gemacht haben, eine sehr große Anzahl an Hosts für einen bestimmten Angriff zu gewinnen. Angreifer gehen heute oft mit legitimen, ungefälschten Pakete vor. Wenn eine Million unterschiedlicher Rechner im Netz ein einziges TCP-SYN-Paket senden, spielt es keine Rolle, dass sie alle ihre reale Absenderadresse verwenden.

4.1.2 Traceback and Pushback

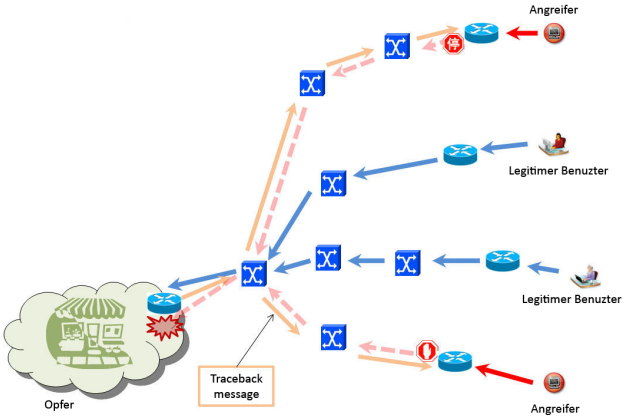


Abbildung 3: Traceback

Es wurden mehrere Methoden einen Angriff zur Quelle zurück zu verfolgen entwickelt. Traceback konzentriert sich auf die Identifizierung der Hosts, die für einen Angriff verantwortlich sind, unternimmt aber ebenso wie Source-Filterung wenig, um den Angreifer am Senden von Datenverkehr zu hindern (siehe Abbildung 3). DoS-Angriffe werden in der Regel von einer (möglicherweise sehr großen) Zahl an kompromittierten Rechnern aus gestartet. Traceback Mechanismen können bei der Ermittlung solcher kompromittierter Hosts von unschätzbarem Wert sein. Dabei erstellt das angegriffene Ziel eine Signatur der Pakete, die den Angriff verursachen. Diese Signatur wird allen dem Ziel vorgelagerten Routern mitgeteilt. So können diese überprüfen, woher der böswillige Datenstrom kommt, diesen zurückverfolgen und selbst die Signatur an benachbarte Router weiterleiten. So kann der Ursprung des Angriffs ermittelt werden. Das aber ist zu spät und den Angriff zu verhindern, oder den initiierenden Täter des selbigen zu ermitteln (wenn der Angriff von einem Bot-Netz ausgeht).

Negativ fällt auf, dass die Implementierung des Systems Performanceeinbrüche verursacht. Außerdem ist man auf die Unterstützung der Internet Service Provider angewiesen, da das Filter-System erst auf den im Internet verbreiteten Router installiert werden muss.

Um diese Einschränkung zu umgehen wurden die beteiligten Traceback-Router mit Filterfunktionalitäten ausgestattet. Mit Pushback (siehe Abbildung 4) kennzeichnet ein Knoten, welche Art von Paketen den Angriff verursacht, und sendet Anfragen an vorgelagerte Knoten deren Paketrate bereits näher an der Quelle zu verringern. Obwohl sich die heutigen Pushback-Vorschläge darauf konzentrieren die Bandbreite einer Verbindung während eines Flooding-Angriffs zu kontrollieren, könnte theoretisch jeder beliebige Host im Internet dynamisches Pushback verwenden, um die Erschöpfung seiner Ressourcen zu verhindern.

Leider kann es schwierig sein Filter zu entwerfen, die perfekt zwischen gewolltem und ungewolltem Traffic unterscheiden. Einstufung anhand der Paket-Header ist anfällig für Spoofing. Einstufung anhand des Paket-Inhaltes wird vereitelt durch die zunehmende Verwendung von Ende-zu-Ende Verschlüsselung. Zudem wird durch die Integration von Filtern eine Grundlage für Zensur geschaffen.

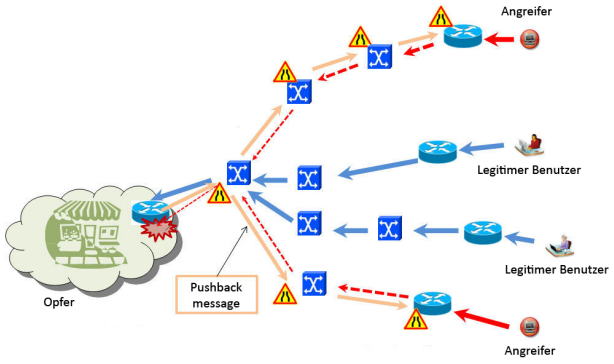


Abbildung 4: Pushback

Ausgeklügelte Angriffe können auch Sonden einsetzen um die Filterfunktion herauszufinden und sie dann zu umgehen. Beispielsweise begrenzt die erste Pushback-Implementierung einfach die Rate aller Pakete mit demselben Ziel. Dieser Ansatz wird nicht nur ebenso gut böswillige Pakete blockieren, er kann auch leicht durch einen Angriff umgangen werden bei denen Pakete mit unterschiedlichem Ziel über denselben Knoten geroutet werden, welcher dann den Flaschenhals darstellt. [5]

4.1.3 Overlay Filtering

Overlays (siehe Abbildung 5) wurden angesichts der hohen Dauer um anspruchsvolle Filter auf der Router-Hardware hinzuzufügen als ein Konzept vorgeschlagen, das es ermöglicht DoS-Filtering Schrittweise zu implementieren. Hier wird der komplette Traffic an ein angegriffenes Ziel über spezielle Zwischenknoten geroutet. Da diese nicht auf dem normalen Routingpfad liegen, können besondere Analyse- und Filtermethoden angewandt werden.

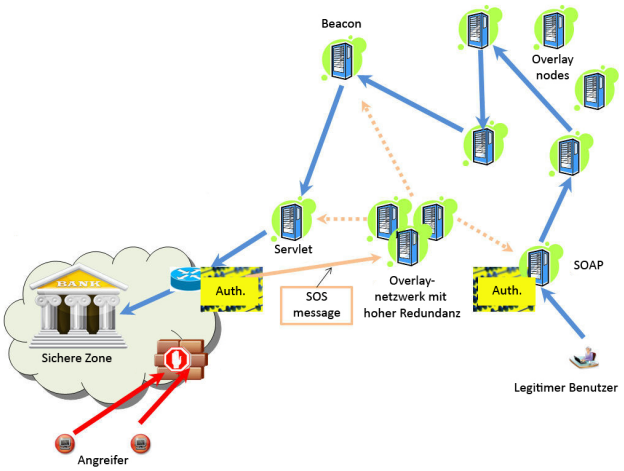


Abbildung 5: Overlay Filtering

Das zu erreichende Ziel (z.B. eine sicherheitskritische Website) wird dabei über einen geheimen Proxy an das Netzwerk angebunden, dem es Dienste zur Verfügung stellen soll. Der Weg zu diesem Proxy führt über ein großes Netz an Overlay-Knoten (Redundanz, falls Pfade im Netz wegen z.B. DDoS nicht erreichbar sind). Der Nutzer der das Overlay-Netzwerk verwenden will, muss sich an einem Einstiegspunkt authentifizieren. Von dort wird er dann an die Overlay-Knoten weitergeleitet. Diese finden den Weg zum Proxy anhand einer geheimen Information im Protokollheader. Router die auf dem Weg zum Ziel liegen können dann so konfiguriert werden, dass alle Pakete, die keine geheime Information beinhalten verworfen werden.

Overlay Filtering hat den Vorteil, dass der Zugriff auf das Netz nur nach stattgefundener Authentifizierung möglich ist. Allerdings sind Angriffe auf die Einstiegspunkte möglich. Wird der Proxy bekannt sind auch dort Angriffe möglich.[1]

4.1.4 Anomaly Detection

Der vielleicht aktivste Bereich in der DoS-Prävention ist Anomalie-Erkennung. Regelbasierte oder statistische Verfahren werden verwendet, um Traffic-Muster in freundlich oder böswillig einzuteilen. Nur eine automatische Reaktion auf einen DoS-Angriff oder eine Sicherheitsverletzung kann schnell genug sein, um Beschädigungen oder Verlust des Dienstes zu verhindern.

Es bestehen allerdings große Bedenken über die Folgen der Implementierung solcher Systeme. Letztlich ist Anomalie-Erkennung keine ausreichende Antwort auf das Problem - die Entscheidung, ob ein bestimmter Datenfluss einen Angriff darstellt oder nicht, muss an den Verbindungsendpunkten auf Anwendungsebene erfolgen.

Schlimmer noch, führt die Anomalie-Erkennung zu abgeschlossenen Systemen, denn ISPs und Systemadministratoren werden sämtliche Traffic-Formen, die nicht standardisiert sind blockieren, um den Wettlauf mit dem Angreifer zu gewinnen. Da Filter-Policies in der Regel geheim sind werden viele Entwickler legitimer Anwendungen nie genau wissen, warum der Verkehr einen Alarm ausgelöst hat beziehungsweise zurückgewiesen wurde.

Wie kann z.B. eine Applikation wissen, dass ein Low-End-Router Ping-Pakete nur bis zu einer bestimmten Rate akzeptiert bevor eine Überlastsituation eintritt, wenn der Router keine Möglichkeit hat, den Absender über seine Ressourcen-Limits zu informieren?

Die Einführung neuer Netzdienste verkäme praktisch zu einer Unmöglichkeit: eine neue Anwendung müsste ihr Datenaufkommen außerordentlich konservativ bemessen, um eine unverhältnismäßige Reaktion von Filter-Systemen oder Netzwerkadministratoren zu vermeiden.[2]

4.2 Präventive Methoden

4.2.1 Client-Puzzles

Client-Puzzles (siehe Abbildung 6) sind eine in der Entwicklung befindliche, viel versprechende Technik, die Service-Garantie für den rechtmäßigen Nutzer ermöglichen will. Diese erhalten den Zugang zu einer Dienstleistung erst, nachdem sie ihre Legitimität nachgewiesen haben. Für jede Service-Anfrage, ist der Nutzer gezwungen, ein kryptographisches Rätsel zu lösen, bevor der Server seine Ressourcen freigibt. Dies stellt eine große Aufgabe für den Angreifer dar, wenn er Traffic in großen Mengen generieren will.

Die Grundidee hinter Client-Puzzles ist, dass jeder Client der den Dienst des Servers ersucht, einige seiner eigenen Ressourcen (Rechenzeit oder Speicher, etc.) verwenden muss, bevor der Server seine Ressourcen für die Verbindung zugesteht. Dies schützt vor Angriffen, die von einer großen Anzahl von Botnetz Computern inszeniert werden, welche echte IP-Adressen verwenden, da die bestehenden DDoS-Tools so sorgfältig konstruiert sind, dass sie den Zombie-Computer nicht zu stören versuchen, um den Eigentümer nicht auf ihre Anwesenheit aufmerksam zu machen.

Dieser Ansatz zur DoS Prävention scheint interessant zu sein, nicht nur aufgrund der formalen Modellierung, sondern auch aufgrund des Protokoll-Designs. Es ist nicht erforderlich, dass Angriffssignaturen und verdächtiger Datenverkehr durch die beteiligten Router und Filter erkannt werden. Da es sich um einen präventiven Ansatz handelt, wird kein legitimer Datenverkehr unterbunden werden. Dies ist auf die Verwendung von Client-Puzzles zurückzuführen. Die Fähigkeit, Rätsel zu lösen, trennt berechtigte Nutzer von automatischen Angriffs-Tools.

Client-Puzzles sind ein Mechanismus, um präventiv gegen DoS-Angriffe vorzugehen. Es ist den Clients möglich, für den Dienst des Servers zu bieten. Die geschieht durch die Berechnung von Rätseln mit unterschiedlichen Schwierigkeitsgraden. Der Server passt den Schwierigkeitsgrad abhängig von der aktuellen Auslastung an. Bei einer Anfrage kann der Client diese entweder akzeptieren, oder er sendet eine Absage an den Server.

Die Clients können dann ihre Leistung erhöhen um den Schwierigkeitsgrad des Rätsels zu bewältigen und senden die Anfrage an den Server zurück. Legitimierte Clients bekommen somit Zugriff auf den Server durch die Erhöhung des Schwierigkeitsgrades, während ein Angreifer weniger Anreiz hat das zu tun, denn er würde nicht wollen, dass der Besitzer des

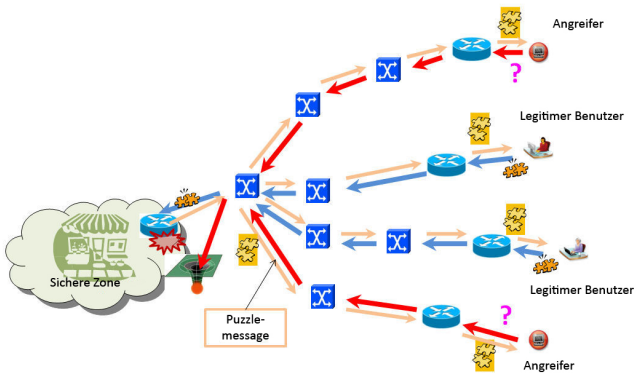


Abbildung 6: Client-Puzzles

befallenen Rechners durch übermäßigen Rechenaufwand benachrichtigt wird.

Ein Client Puzzle könnte etwa ein Java-Script in einer Website sein, dass aus einem gegebenen Hashwert per Brute-Force Methode ein verschlüsseltes Secret berechnen sollte. Über den Grad der Verschlüsselung lässt sich der Schwierigkeitsgrad bestimmen.

Standard Client-Puzzle-Protokolle funktionieren gut um das Aufbrauchen von Ressourcen auf dem Server bei Angriffen zu verhindern. Sie scheitern jedoch, wenn der Angreifer direkt eine riesige Welle an Paketen an einen bestimmten Server sendet, um dessen Bandbreite zu erschöpfen. Um diesen bösartigen Verkehr zu blockieren, muss ein Filter-Mechanismus in den dazwischen liegenden Netzwerken verwendet werden, anstatt auf Server-Ebene.

5. ZUSAMMENFASSUNG

Wie nun gezeigt wurde, besteht ein ständiger Wettkampf zwischen organisierten Kriminellen beim Finden neuer Angriffsmethoden und den Entwicklern geeigneter Gegenmaßnahmen. Diese Maßnahmen sind leider auch immer mit der Einschränkung des Komforts des Benutzers bzw. mit Einbußen bei der Einführung und Administration von Netzdiensten verbunden.

Um auf die vielfältigen Angriffsmöglichkeiten wie Ressourcen- und Bandbreitenausschöpfung, SYN-Floods, ICMP-Floods, IP-Spoofing und Botnetze reagieren zu können genügt es nicht einen einzigen Lösungsansatz zu verfolgen.

Es stehen mit den gegebenen Filter-Methoden, mit Traceback- und Pushback-Ansätzen, mit Overlay-Netzwerken einige hilfreiche Werkzeuge zur Verfügung, um Angriffe abzuwenden. Besonders der Bereich Anomalieerkennung erfährt zur Zeit große Beachtung in der Wissenschaft.

Vielversprechend sind die neuen, sich in der Entwicklung befindlichen spieletheoretischen Ansätze, bei denen zwischen Client und Server eine Nutzenfunktion optimal erfüllt werden muss, damit Zugriff gewährt wird oder nicht. Client-Puzzles stellen im Moment eine aussichtsreiche Lösung dar, die das Resource-Exhaustion Problem beseitigen könnte.

Leider kann keine der aufgeführten Lösungen alle Herausforderungen lösen. Für ISPs und Administratoren gilt es daher, eine geeignete Kombination der vorhandenen Werkzeuge einzusetzen. Die Kaskadierung der einzelnen Lösungen scheint angebracht. So sollte grundlegende Sicherheit durch hoch verbreitete Filter-Methoden schon in den ISP-Netzen gegeben werden. Auch muss die Unterstützung für Traceback und Pushback Möglichkeiten sowie Anomalieerkennung gegeben sein. Um den Benutzer so wenig wie möglich zu beeinträchtigen sollten weitere Methoden wie z.B. Client-Puzzles allerdings erst (und dies natürlich automatisiert) zum Einsatz kommen, wenn ein akuter Angriff vorliegt.

6. LITERATUR

[1] A. D. Keromytis, V. Misra, and D. Rubenstein. Sos: secure overlay services. In *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 61–72, New York, NY, USA, 2002. ACM.

[2] A. Mahimkar and V. Shmatikov. Game-based analysis of denial-of-service prevention protocols. In *CSFW '05: Proceedings of the 18th IEEE workshop on Computer Security Foundations*, pages 287–301, Washington, DC, USA, 2005. IEEE Computer Society.

[3] J. Mirkovic and P. Reiher. A taxonomy of ddos attack and ddos defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 34:39–53, 2004.

[4] Wikipedia. Denial of service. Website, 2010. Available online at `http://de.wikipedia.org/w/index.php?title=Denial_of_Service&oldid=75812106` visited on June, 25th of 2010.

[5] J. Xu and W. Lee. Sustaining availability of web services under distributed denial of service attacks. *IEEE Trans. Comput.*, 52(2):195–208, 2003.

Stärken und Schwächen von PKI

Johanna Cuno
Betreuer: Ralph Holz
Seminar Innovative Internettechnologien und Mobilkommunikation SS2010
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: cunoj@in.tum.de

KURZFASSUNG

Public Key Infrastructure (PKI) basiert auf den Methoden der asymmetrischen Kryptographie. Der Ansatz besitzt mehrere Vorteile, die ein Grund für die große Verbreitung von PKI sind. So unterstützt die Technologie wichtige Sicherheitskriterien wie Authentifizierung, Vertraulichkeit, Integrität und Nicht-Abstreitbarkeit. Zudem ermöglicht die am häufigsten eingesetzte PKI-Variante, die hierarchische PKI, eine zentral organisierte Administration von Zertifikaten bzw. Schlüsseln. Den Vorteilen steht eine Reihe von Nachteilen gegenüber. Ein ungelöstes Problem betrifft die Handhabung der Zertifikatsperrung. Weitere Probleme beruhen darauf, dass die der PKI zu Grunde liegenden Modelle Vertrauensbeziehungen (zum Beispiel zwischen Nutzer und Zertifizierungsstelle) modellieren, die mit den Realwelt-Bedingungen schwer vereinbar sind. Zusammengenommen überwiegen die Schwierigkeiten, die bei der Umsetzung von PKI in die Praxis auftreten.

Schlüsselworte

Public Key Infrastructure (PKI), digitales Zertifikat, digitale Signatur, Zertifizierungsstelle (engl. Certificate Authority, CA), Vertrauensmodelle, Verschlüsselungsverfahren

1. EINLEITUNG

Im globalen Wettbewerb ist es für Unternehmen von existenzieller Bedeutung, sicher kommunizieren und elektronische Transaktionen durchführen zu können. Um eine sichere Kommunikation im Unternehmensnetzwerk, aber ebenso mit Business-Partnern, Lieferanten und Kunden gewährleisten zu können, ist eine verlässliche IT-Infrastruktur notwendig. Gleichermaßen besteht ein sehr hoher Sicherheitsbedarf bei staatlichen Institutionen.

Public Key Infrastructure (PKI) wird weithin als die IT-Technologie betrachtet, die diesen Sicherheitsanforderungen gerecht wird ([1], [2]). In vielen Organisationen ist PKI zentraler Bestandteil der Sicherheitsarchitektur. Es gibt jedoch auch einige kritische Stimmen, die Schwächen von PKI aufzeigen und teilweise den gesamten Ansatz in Frage stellen [4]. Beides, Vor- und Nachteile des PKI-Ansatzes, sind Inhalt dieser Ausarbeitung.

Zu Beginn werden einige kryptographische Grundlagen behandelt. Dazu werden in Abschnitt 2 die Verfahren der symmetrischen und asymmetrischen Verschlüsselung und das Konzept der digitalen Signatur beschrieben. Im dritten Teil wird das Konzept von PKI erklärt. In den Abschnitten 4 und 5 werden schließlich die wesentlichen Stärken und Schwächen von PKI herausgestellt. Der Artikel endet mit einem kurzen Ausblick zu möglichen Alternativen in Kapitel 6 und einer abschließenden Bewertung in Kapitel 7.

2. GRUNDLAGEN

Im Folgenden werden zunächst die zwei prinzipiellen Arten der Verschlüsselung erklärt und voneinander abgegrenzt. Zudem wird der Begriff der digitalen Signatur behandelt, da dieser für das Verständnis von PKI grundlegend ist.

2.1 Symmetrische Kryptographie

Bei symmetrischen Verschlüsselungsverfahren wird ein geheimer Schlüssel verwendet. Auf der Seite des Senders wird das Dokument mit einem geheimen Schlüssel verschlüsselt. Der Empfänger kann die verschlüsselte Nachricht nur mittels desselben geheimen Schlüssels entschlüsseln. Dazu müssen die beteiligten Parteien einen sicheren Weg finden, den geheimzuhaltenden Schlüssel miteinander auszutauschen. Dies ist der wesentliche Nachteil der symmetrischen Verschlüsselung. In [2] werden diese Problematik und ein möglicher Lösungsansatz (Key Distribution Center, KDC) diskutiert.

Der große Vorteil symmetrischer Verschlüsselungsverfahren ist, dass sie im Vergleich zu den nachfolgend beschriebenen asymmetrischen Verfahren einen geringeren Rechenaufwand und kürzere Schlüssel erfordern.

In Abbildung 1 ist das Verfahren bei der symmetrischen Verschlüsselung veranschaulicht.

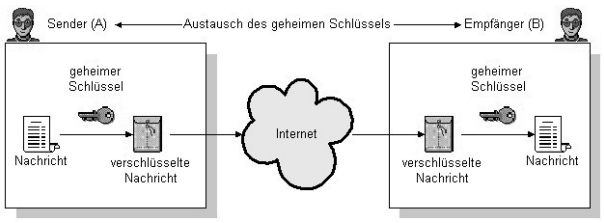


Abbildung 1: Symmetrische Verschlüsselung [11]

2.2 Asymmetrische Kryptographie

Bei den Verfahren der asymmetrischen Verschlüsselung (engl. public key encryption) verschlüsselt der Sender ein Dokument mit dem öffentlichen Schlüssel des Empfängers. Dieser wiederum entschlüsselt die Nachricht mit seinem privaten

Schlüssel (vgl. Abb. 2). Das Problem, dass ein geheimer Schlüssel ausgetauscht werden muss, entfällt damit.
Die Verteilung der öffentlichen Schlüssel ist einfach zu handhaben. Einziger Nachteil dieser Public-Key Verschlüsselungsverfahren ist, dass sie mit einem höheren Rechenaufwand verbunden sind und längere Schlüssel benötigen.
Die Methoden der asymmetrischen Kryptographie werden zudem auch zum digitalen Signieren verwendet.

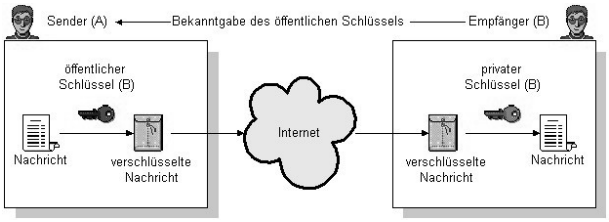


Abbildung 2: Asymmetrische Verschlüsselung [12]

2.3 Digitale Signaturen

Eine digitale Signatur hat - ähnlich wie eine handschriftliche Unterschrift - den Zweck, ein Dokument so zu kennzeichnen, dass der Ersteller des Dokuments eindeutig identifiziert und die Urheberschaft des Dokuments nicht abgestritten werden kann. Beim Signieren wird auf die Daten zunächst eine Hashfunktion angewandt. Die gehashten Daten werden mit einem privaten Schlüssel verschlüsselt. Diese Signatur wird dem Dokument angehängt und beides zusammen wird dem Empfänger übermittelt (vgl. linke Seite der Abbildung 3).
Auf der Seite des Empfängers wird die Signatur mit dem dazugehörigen öffentlichen Schlüssel entschlüsselt. Ist das Ergebnis dieses Verfizierungsprozesses derselbe Hashwert wie derjenige des ursprünglichen Dokuments, wurden die Daten während der Übermittlung nicht verändert (vgl. rechte Seite der Abbildung 3). Der Empfänger kann sich der Authentizität des Senders sicher sein und dass die Daten tatsächlich von dem angegebenen Sender stammen und unverfälscht sind.

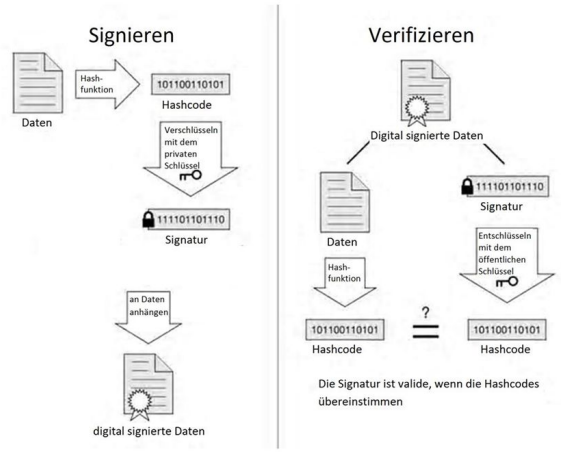


Abbildung 3: Digitales Signieren und Verifikation

3. WAS IST PKI?

Public Key Infrastructure basiert auf den Verfahren der asymmetrischen Kryptographie. Das Problem dieser Verfahren *ohne* zusätzliche Infrastruktur ist, dass beim Austausch des öffentlichen Schlüssels die Gefahr einer Man-in-the-middle-attack (MITM) besteht. Wenn zum Beispiel Person A ihren öffentlichen Schlüssel an Person B sendet und es einem Angreifer gelingt, den öffentlichen Schlüssel dabei abzufangen, kann dieser unbefugte Dritte den richtigen Schlüssel durch seinen eigenen ersetzen. Der Sender B verwendet dann beim Verschlüsseln unwissentlich den öffentlichen Schlüssel des Man-in-the-middle. Der Angreifer hat dann die Möglichkeit, die Nachricht von B an A mit seinem privaten Schlüssel zu entschlüsseln, zu lesen und sie anschließend mit dem richtigen Schlüssel von A zu verschlüsseln und an A weiterzuleiten. Das zentrale Problem hierbei ist, dass man einem öffentlichen Schlüssel per se nicht ansieht, zu wem er gehört. Die Folge ist, dass weder Person A noch Person B die MITM bemerken.
An dieser Stelle setzt das Konzept der PKI und digital signierter Zertifikate an. Ein digitales Zertifikat ist eine digital signierte Datenstruktur, die einen öffentlichen Schlüssel an die Identität seines Besitzers bindet. Der Besitzer kann dabei sowohl eine Einzelperson als auch ein Unternehmen oder eine Anwendung sein. Die digitale Signatur dient dem Nachweis der Authentizität des Zertifikateigentümers. Auf einem Zertifikat (vgl. Abb. 4) können zusätzliche Informationen wie zum Beispiel Gültigkeitsdauer und Verwendungszweck gespeichert werden.

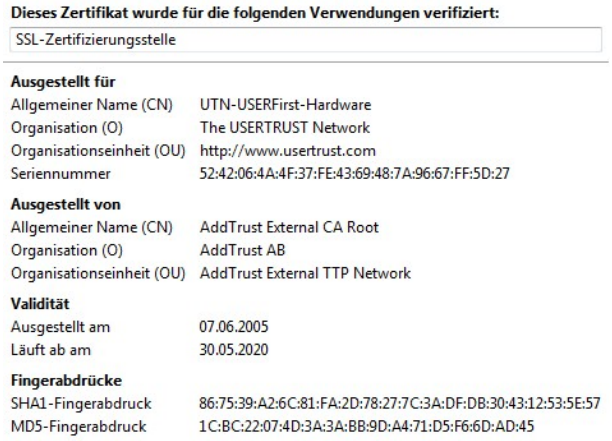


Abbildung 4: Digitales Zertifikat

PKI ist eine Infrastruktur, die sich aus verschiedenen Technologien, Verfahren und den beteiligten Anwendern zusammensetzt. Sie umfasst ein System von Hardware- und Softwarekomponenten. Wesentliche Elemente hierbei sind die Client-Software auf der Seite des Anwenders und auf der Serverseite ein *Certificate Repository*, das einen schnellen Zugriff auf gesuchte Zertifikate ermöglicht. Weiterer Bestandteil einer PKI sind sämtliche Methoden zur Erstellung, Verwaltung, Verteilung und Sperrung digitaler Zertifikate. Diese sind zumeist in einer sogenannten Policy festgelegt und variieren in Abhängigkeit des zugrundeliegenden

(Vertrauens-)Modells (vgl. nächster Abschnitt). Ausgestellt werden digitale Zertifikate von Zertifizierungsstellen (engl. Certificate Authority, CA). Eine CA kann von einer Behörde oder einem Unternehmen betrieben werden.
Es existieren verschiedene Modelle, wie eine PKI in der Praxis umgesetzt werden kann. Diese Modelle haben damit zu tun, welchem Zertifikat bzw. welcher Zertifizierungsstelle ein Benutzer vertrauen kann. Man spricht deshalb auch von *Vertrauensmodellen* (engl. trust models). Das gängigste Konzept ist das hierarchische Vertrauensmodell. Es sieht eine Wurzelinstanz, die Root CA, vor, die entweder vermittelt über weitere, untergeordnete Zertifizierungsstellen oder auf direktem Weg dem Endnutzer Zertifikate ausstellt (vgl. Abb. 5).

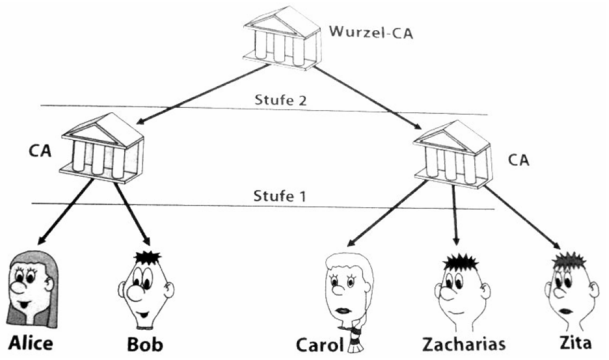


Abbildung 5: Hierarchisches Vertrauensmodell [5]

Wenn sich zwei CAs gegenseitig zertifizieren, spricht man von *Cross Certification*. Wie Abbildung 6 zeigt, können diese beiden Zertifizierungsstellen ihrerseits hierarchisch organisiert sein. Eine Cross Certification ist zum Beispiel sinnvoll, wenn Unternehmen fusionieren, die jeweils eigene Zertifizierungsstellen betreiben.

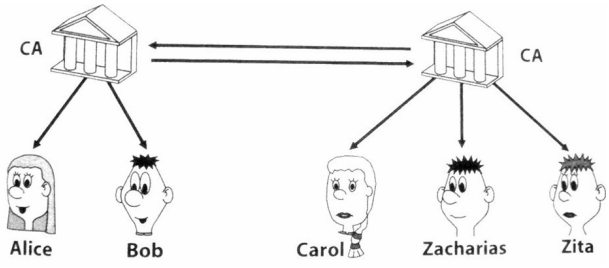


Abbildung 6: Cross Certification [5]

Ein anderer, dezentraler Ansatz der Vertrauensmodellierung wird mit dem so genannten *Web of Trust (WoT)* verfolgt. Die Grundidee ist, dass das Signieren und Ausstellen von Zertifikaten nicht mehr von zentralen Instanzen wie Certificate Authorities übernommen wird. Vielmehr kann jeder Benutzer selbst Zertifikate ausstellen und darüber individuell entscheiden, welchen potentiellen Kommunikationspartnern er vertraut. Vertrauensketten und ein "Netz des Vertrauens" entstehen, wenn Benutzer nicht nur den Personen, die sie persönlich kennen, vertrauen, sondern darüber hinaus auch deren Vertrauenspartnern (vgl. Abb. 7). Gleichwie im Modell der hierarchischen PKI wird auch im Web of Trust eine Transitivität des Vertrauens modelliert.

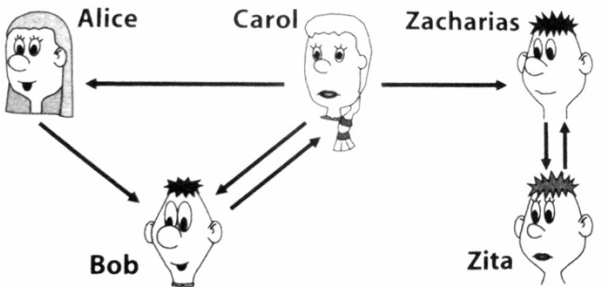


Abbildung 7: Web of Trust [5]

4. STÄRKEN VON PKI

Ein wichtiger Vorteil von PKI ist, dass das Konzept die wesentlichen Sicherheitskriterien Authentifizierung, Integrität und Vertraulichkeit unterstützt. Durch die Verwendung digitaler Signaturen wird außerdem das Kriterium der Nichtabstreitbarkeit sichergestellt, da die Urheberschaft signierter Nachrichten nicht abgestritten werden kann. In [2] sind diese Sicherheitsziele näher beschrieben.
Weiterhin erfolgt bei PKI eine zentrale Administration der öffentlichen Schlüssel. Dies ist insbesondere bei firmeninterner Umsetzung von PKI bedeutsam, da hierdurch Komplexität und Verwaltungsaufwand verringert werden.
Damit einher geht der Aspekt, dass im Rahmen einer zentral gesteuerten PKI eine Policy (vgl. z. B. [8]), die zum Beispiel Regeln zur Erneuerung und Sperrung von Zertifikaten festlegt, vergleichsweise einfacher durchzusetzen ist als beispielsweise in einem Web of Trust.

5. SCHWÄCHEN VON PKI

Bei der Umsetzung von PKI in die Praxis existieren mehrere Probleme. Neben zwei allgemeinen Schwierigkeiten bei der Anwendung von PKI, werden im Folgenden Probleme im Zusammenhang mit der Sperrung von Zertifikaten und der Rolle von Vertrauen bei der Anwendung von PKI erläutert.

5.1 Allgemeine Probleme

Eine der Schwächen von PKI ist, dass sich die Implementierung oft weit schwieriger gestaltet als vielfach propagiert wird [3]. So ist die Einrichtung von PKI meist mit aufwendigen Änderungen der Altsysteme verbunden. Zudem müssen die Nutzer ausgiebig geschult werden und zumindest am Anfang entsteht bei der Verteilung großer Mengen von Zertifikaten ein beträchtlicher Verwaltungsaufwand. Johnson und Johnson [6] berichten zum Beispiel davon, dass allein im ersten Jahr nach Einführung von PKI bei einer Mitarbeiterzahl von 110.000 insgesamt 30.000 Zertifikate gesperrt werden mussten. Diese hohe Zahl wird vor allem auf die unzureichende Schulung der Mitarbeiter im Umgang mit den Zertifikaten zurückgeführt.
Wie bei allen anderen Technologien, gilt auch für PKI, dass

ein System nur so sicher ist wie seine schwächste Komponente. Sowohl auf Seiten der beteiligten Computersysteme als auch auf Seiten der Nutzer können Sicherheitslücken entstehen. Es ist zum Beispiel denkbar, dass ein privater Schlüssel nicht ausreichend geschützt wird. Wird er gestohlen oder kopiert, können Dokumente unrechtmäßig signiert oder entschlüsselt werden.

5.2 Sperrung von Zertifikaten

Eine wichtige Anforderung an eine PKI ist, dass bereits ausgestellte Zertifikate gesperrt werden können. Dies ist zum Beispiel notwendig, wenn sich die Identität eines Zertifikatbesitzers ändert (z. B. Namenswechsel), ein Mitarbeiter das Unternehmen verlässt oder ein privater Schlüssel gestohlen oder kopiert wird. Eine unmittelbare Sperrung des entsprechenden Zertifikats ist dann von großer sicherheitskritischer Bedeutung.
Entsprechend wichtig ist es, dass sich die Nutzer einer PKI über den Status von Zertifikaten informieren können. Eine PKI muss also einen Mechanismus zur Verfügung stellen, der die Überprüfung von Zertifikaten auf ihre Gültigkeit gestattet. Dieser Vorgang wird als *Revocation Check* bezeichnet. Ein vielfach praktizierter Ansatz ist eine Art Blacklist mit gesperrten Zertifikaten, die *Certificate Revocation List (CRL)*, die in bestimmten Zeitabständen aktualisiert wird und dann durch den Nutzer von einem Server heruntergeladen werden kann.
Nach Gutmann [4] ist diese Lösung unzureichend. Das zentrale Problem dabei ist, dass CRLs nur in bestimmten Zeitabständen aktualisiert werden. Der Nutzer erhält keine Information darüber, wenn in der Zeit zwischen zwei Updates ein Zertifikat gesperrt wurde. Entsprechend problematisch ist dies, wenn ein gesperrtes Zertifikat nicht als solches erkannt wird. Notwendig wäre eine Echtzeit-Statusabfrage, welche aber mit CRLs nicht realisierbar ist. Ein zusätzliches Problem ist, dass CRLs sehr umfangreich werden können. Je häufiger eine CRL herausgegeben wird, desto größer die Netzwerk- und Serverbelastung.
Eine mögliche Alternative zu CRLs bietet OCSP (Online Certificate Status Protocol). Es erlaubt dem Nutzer, bei einem Server, dem sogenannten OCSP-Responder den Status eines Zertifikats abzufragen. Obwohl OCSP der Lösung mit CRLs überlegen ist, gibt es auch bei diesem Ansatz einige Probleme. Kritisiert wird zum Beispiel, dass der OCSP-Responder unpräzise und nicht eindeutige Informationen zum Status eines Zertifikats liefert [4]. Mögliche Antworten sind "gesperrt", "nicht gesperrt" und "unbekannt". So können sich hinter der Antwort "unbekannt" mehrere Bedeutungen verbergen. Sie wird dem Anwender übermittelt, wenn das betreffende Zertifikat nicht ausgestellt wurde, aber ebenfalls, wenn es ausgestellt, aber nicht abrufbar ist. Das Ergebnis "nicht gesperrt" bedeutet lediglich, dass das Zertifikat aktuell nicht gesperrt ist. Es beinhaltet nicht zwangsläufig, dass das Zertifikat auch valide ist, weil Kritierien wie Gültigkeitsdauer oder Verwendungszweck nicht in die Überprüfung miteinbezogen werden. Dies ist das Hauptproblem, das Gutmann [4] im Zusammenhang mit CRLs sieht und das bei OCSP gleichermaßen besteht. Beide Instrumente liefern ausschließlich Informationen darüber, ob ein Zertifikat gesperrt ist. Die Information, die der Anwender tatsächlich benötigt, ist, ob das Zertifikat auch valide ist. Für weitere Informationen zu den Vor- und Nachteilen von OCSP sei auf [4] verwiesen.

5.3 Prinzip des Vertrauens

Das Prinzip des Vertrauens und die verschiedenen Vertrauensmodelle sind für das Konzept von PKI grundlegend. Allerdings gehen damit eine Reihe weiterer Probleme einher. Bei einer hierarchischen PKI genießt die Zertifizierungsstelle an der Spitze der Hierarchie, die Root CA, unbegrenztes Vertrauen, da sie von keiner anderen CA bestätigt werden muss.
Der Nutzer an der Basis muss der Root CA in zweierlei Weise vertrauen. Zum einen muss er sich darauf verlassen, dass sie sichere und valide Verfahren beim Ausstellen von Zertifikaten verwendet, das heißt eine sorgfältige Identitätsprüfung vornimmt. Darüber hinaus muss er darauf vertrauen, dass die Root CA korrekt bei der Validierung untergeordneter oder anderer gleichberechtigter CAs (Cross Certification) vorgeht. Insbesondere bei einer mehrstufig hierarchischen PKI, aber ebenso in einem Web of Trust, wird eine Transitivität des Vertrauens modelliert, die in der realen Welt aber in keiner Weise gegeben ist. Wichtig ist, dass dies kein eigentliches Problem von PKI ist. Vielmehr bilden die Vertrauensmodelle das Geschehen in der realen Welt in ungeeigneter Weise ab.

In der Praxis kann dies erhebliche negative Folgen nach sich ziehen. Bei der Nutzung von Webdiensten, die eine verschlüsselte Kommunikation erfordern, wie zum Beispiel E-Mail-Dienste oder Online-Bankingsysteme, zeigt der Browser entweder eine Sicherheitswarnung oder ein gelbes Sicherheitsschloss an. Letzteres signalisiert dem Nutzer, dass die Seite vertrauenswürdig ist und die Kommunikation tatsächlich verschlüsselt erfolgt. Tatsächlich bedeutet das Symbol aber lediglich, dass eine Root CA oder eine untergeordnete, von der Root CA validierte CA diesen Kommunikationspartner authentifiziert hat. Ob diese CA wiederum vertrauenswürdig ist, entscheiden die Softwarehersteller. Diese entscheiden, welche Root-Zertifikate in den Browser, in den sogenannten Truststore, aufgenommen werden. Das bedeutet, dass nicht der Nutzer selber, sondern der Webbrowser darüber entscheidet, ob in die Identität eines Kommunikationspartner im WWW vertraut wird oder nicht.

5.4 Szenario

Die meisten Webbrowser sehen ein Aufnahmeverfahren für Root-Zertifizierungsstellen vor, das die Vertrauenswürdigkeit der Antragsteller überprüft (z. B. [9]). Es ist jedoch ein Szenario denkbar, nach dem einmal vom Browser als "trusted" eingestufte Root CAs ihre Macht irgendwann für kriminelle Zwecke nutzen [7]. Mittels gefälschter Zertifikate könnte der E-Mailverkehr von Nutzern abgehört oder Wirtschaftsspionage betrieben werden.
Ebenso ist es vorstellbar, dass in einem Knotenpunkt auf dem Pfad zwischen dem Rechner des Anwenders und einem Webserver (z. B. auf einem Heimrouter) neben einem echten Zertifikat ein weiteres falsches Zertifikat installiert wird. Der Webnutzer würde zum Beispiel mit Googlemail über ein falsches gmail-Zertifikat kommunizieren. Daneben läge das richtige Zertifikat, über das Google kommunizieren würde. Die gesamte Kommunikation könnte abgehört werden, ohne dass Google oder der User diese Man-in-the-middle-attack bemerken würden (vgl. Abb. 8).

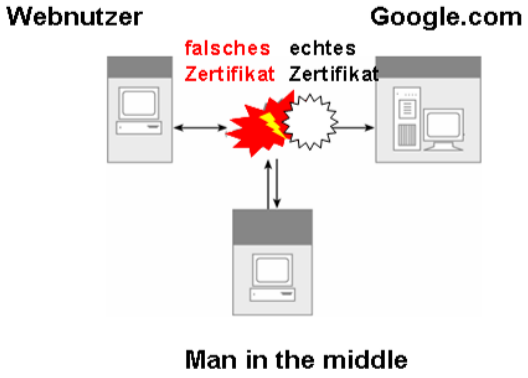


Abbildung 8: Man-in-the-middle-attack Szenario

6. ALTERNATIVEN

Vor dem Hintergrund der genannten Probleme bei der Anwendung von PKI existieren viele alternative Lösungsansätze. Nach Schmeh [5] gelten Identitätsbasierte Krypto-Systeme derzeit als die wichtigste Alternative zu PKI.
Vielfach wird auch das bereits in Abschnitt 3 beschriebene Web of Trust als bessere Variante diskutiert (vgl. z. B. [13]). Die bekanntesten Umsetzungen des Web of Trust in der Praxis ist die kommerzielle Software Pretty Good Privacy (PGP) und das OpenSource-Programm GNU Privacy Guard (GnuPG).
Das Web of Trust gilt als deutlich flexibler als die hierarchische PKI, da der Benutzer über deutlich mehr Handlungsspielraum und individuelle Kontrolle verfügt. Mit der Anwendung des WoT im WWW sind jedoch noch Probleme verbunden. Zum einen muss ein Benutzer die Vertrauenswürdigkeit und Identität potentieller Kommunikationspartner nun selbst überprüfen. Ist der betreffende Kommunikationspartner ein Webdienst im WWW, ist diese Überprüfung prinzipiell schwieriger, als wenn sie beispielsweise in einem persönlichen Treffen zwischen zwei Personen erfolgt. Zum anderen bedeuten die sehr großen Nutzerzahlen im WWW, dass das Durchsetzen von Policies nur schwer zu erreichen ist [5] und die Koordination und Administration der beteiligten Anwender eines Web of Trust schwierig zu handhaben ist [13]. In begrenzten Umgebungen allerdings, zum Beispiel im Email-Verkehr oder in Unternehmensnetzwerken hat sich das Web of Trust bereits als erfolgreiches Konzept erwiesen [10].

7. ZUSAMMENFASSUNG

Die wesentlichen Vorteile von PKI sind die Unterstützung wichtiger Sicherheitskriterien und die Reduktion von Komplexität. Die Authentifizierung und die Beweiskraft mittels digital signierter Zertifikate sind zentral für das Konzept von PKI. Positive Erfahrungsberichte gibt es vor allem, wenn PKI in einem begrenzten Rahmen, zum Beispiel unternehmensintern, angewandt wird.
Den Stärken stehen jedoch eine Reihe von ungelösten praktischen Problemen gegenüber. Dies betrifft die Sperrung von Zertifikaten und die gesamte Vertrauensproblematik, die insbesondere im WWW gravierende sicherheitsrelevante Folgen nach sich ziehen kann. Das Web of Trust ist eine interessante Alternative zur hierarchischen PKI, das aber derzeit technisch noch nicht ausgereift ist [10]. Zusammengefasst gibt es daher bislang keine Alternativen, die sich großflächig durchgesetzt haben oder das Potential haben, dies in naher Zukunft zu tun.

8. LITERATUR

[1] S. K. Katsikas, J. Lopez & G. Pernul: *Security, Trust and Privacy in Digital Business*. Second International Conference, TrustBus, Copenhagen, Denmark. Proceedings. Aug. 2005

[2] C. Adams & S. Lloyd: *Understanding PKI: Concepts, Standards and Deployment Considerations*, 2nd ed. Addison-Wesley, 2003

[3] C. Ellison & B. Schneier: Ten risks of PKI: What you're not being told about public key infrastructure. *Computer Security Journal*, 16(1): 1-7, 2000. Online verfügbar unter: `http://www.counterpane.com/pki-risks.html`

[4] P. Gutmann: PKI: It's Not Dead, Just Resting. *Computer*, vol. 35, no. 8, pp. 41-49, 2002. Online verfügbar unter: `http://csdl.computer.org/comp/mags/co/2002/08/r8toc.htm`

[5] K. Schmeh: *Kryptographie. Verfahren - Protokolle - Infrastrukturen*, 3. überarb. Aufl. dpunkt.verlag, 2007

[6] S.W. Smith: Deploying and Using Public Key Technology: Lessons Learned in Real Life. *IEEE Security Privacy*, 2004

[7] C. Soghoian & S. Stamm: *Certified Lies: Detecting and Defeating Government Interception Attacks Against SSL*. Online verfügbar unter: `http://files.cloudprivacy.net/ssl-mitm.pdf`, 2010

[8] *Mozilla CA Certificate Policy*, `http://www.mozilla.org/projects/security/certs/policy/`

[9] Mozilla Wiki: *CA: How to apply*, `https://wiki.mozilla.org/CA:How_to_apply#Applying_for_root_inclusion_in_Mozilla_products`

[10] *Web Security Trust Models*, `http://www.freedom-to-tinker.com/blog/sjs/web-security-trust-models`

[11] A. Lauert: *Sicherheitskonzepte*, `http://ddi.cs.uni-potsdam.de/Lehre/e-commerce/elBez2-5/page05.html`, 2002

[12] A. Lauert: *Sicherheitskonzepte*, `http://ddi.cs.uni-potsdam.de/Lehre/e-commerce/elBez2-5/page06.html`, 2002

[13] G. Caronni: Walking the Web of Trust. *IEEE Computer Society Press*, 2000. Online verfügbar unter: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.21.7392&rep=rep1&type=pdf`

Distributed PKI in P2P Networks

Martin Schanzenbach
Betreuer: Matthias Wachs
Seminar Innovative Internettechnologien und Mobilkommunikation SS2010
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: schanzen@in.tum.de

ABSTRACT
Internet security today is based almost entirely on a public key infrastructure that allows authentication and encryption of data. However this PKI heavily relies on central institutions, namely the Certification Authorities (CAs), that issue certificates. In pure P2P networks such central instances are unwanted because they contradict the P2P paradigms. In this paper, we describe how a Certification Authority can be efficiently maintained and distributed across all peers in a P2P network. This makes it possible to put away with central CAs in P2P networks by establishing a *distributed* public key infrastructure (DPKI).

Keywords
P2P, PKI

1. INTRODUCTION
Today peer-to-peer (P2P) networks are becoming increasingly popular. They provide a lot of features that traditional client and server architectures simply don't, for example there is usually no single point of failure.

In common client and server architectures *authentication* is often realised through a *public key infrastructure* (PKI). PKI authentication today is usually done using X.509 certificates. Servers can identify themselves to the client by showing their certificate which needs to be signed by a *trusted third party*. Those *trusted third parties* are also called *Certification Authorities* (CA) because they have the ability to issue the certificates.

However the CA is a central instance in the network and thus opposes the idea of a pure P2P system. In this paper we try to solve this issue by distributing the functionality of the CA across all peers in a P2P network. In such a *distributed* public key infrastructure (DPKI) collaborating peers provide the functionality of the CA. Hence for example issuing a certificate depends on multiple peers working together. Also the single *trusted third party* mentioned above is represented by all the peers in the network.

This paper is structured as follows: At first we present some related work that has already been done in terms of distributed cryptography. In chapter 3 we discuss the basics of P2P networks. Chapter 4 deals with the cryptographic routines used to realise the distributed PKI. A small introduction of the PKI is given in Chapter 5. Chapter 6 then explains in detail the approach to set up a distributed PKI. Finally in chapter 7 a few possible attacks on this certification scheme are discussed.

2. RELATED WORK
Shamir proposes a method in [5] that allows to divide any data into n parts where the knowledge of k parts is enough to completely reconstruct the original data. Whereas knowing only $k-1$ parts will yield no information on the data whatsoever. The author called this a (k, n) threshold scheme that can be "very helpful in the management of cryptographic keys" [5]. This is particularly interesting in the scope of our work to distribute a public key infrastructure.

Another interesting work was done by Rivest et al. [4]. They present a *ring signature* scheme. This signature can be created by anyone of the same group, or "ring", with his own secret key. The signature can be verified with the group's public key, that is shared across all members. Hence it is not possible to identify the signer, which is an advantage in terms of privacy. However they do not propose a concrete implementation of their scheme.

Finally we are going to look at some related work regarding distributed certification. Zhou et al. [6] propose a distributed *Certification Authority* by the use of *threshold cryptography*. The authors present *Cornell On-line Certification Authority (COCA)*, a "fault-tolerant and secure on-line certification authority". This approach allows a ratio of servers that act as the CA to get compromised. As long as this ratio is within a certain threshold the certification process is still possible because enough servers can collaborate to create the certificate. This is possible due to the use of *threshold cryptography* which is explained in detail in section 4. However this approach only addresses client and server architectures and not P2P systems.

3. BASICS OF P2P NETWORKS
Peer-to-peer (P2P) networks are often used to design high available and low-cost systems. Putting away with the traditional client/server model, in which one system acts as a central instance (the server), the P2P network treats every node in the network equally. This means that participating peers can act as both – client and server – depending on the situation. There are two different kinds of peer-to-peer networks – structured and unstructured ones. Both will be discussed in the following.

3.1 Unstructured

When a peer requests data in an unstructured P2P network its request is flooded through the network. Such a technique creates a very high amount of signalling traffic [**?**]. An example for an unstructured P2P network is Gnutella. Today *structured* P2P networks are more common because they provide the basics for a more efficient network.

3.2 Structured

A structured P2P network allows any peer to efficiently look up data in the network. This is often realised through *distributed hash tables* (DHTs). DHTs serve a similar function like traditional hash tables. Peers can efficiently find data by using the DHT to look up a peer that owns the data. This works by querying the DHT for peers that can provide a specific data set. An example for such a network, that is using Kademlia DHTs, is the distributed tracking system of the popular BitTorrent protocol.

4. BASICS OF DISTRIBUTED ASYMMETRIC CRYPTOGRAPHY

PKI in general is based on asymmetric cryptography. The distributed approach for P2P networks we propose is no different. However the processes of *key generation*, *generation of signatures* and *encryption* differ slightly from the traditional non-distributed approach.

At first we are going to discuss the basics of asymmetric cryptography. Then we take a look at methods for key generation as well as signing and encrypting in a distributed fashion.

4.1 Asymmetric cryptography

Asymmetric cryptography is a method that can be used to sign and encrypt data. It is quite different to its symmetric counterpart. Instead of using a single secret key for encryption so called *key-pairs* (P, S) consisting of a **public** key P and a **private** key S are used. The public key P can be distributed through insecure channels whereas the private key S should always remain secret. To **encrypt** any kind of data **t** the public key can be used. **Decryption** is possible by applying the private key. RSA is a well known cryptosystem that can be used for asymmetric cryptography. It defines $P = (d, m)$ and $S = (e, m)$. In this case **e** is the secret part of the key pair, whereas **d** is the public part. The modulo $\mathbf{m} = \mathbf{p} * \mathbf{q}$ is required for the calculations when de- and encrypting and p and q are big prime numbers. This is why the security of the RSA cryptosystem is largely based on the mathematical problem of factoring large numbers. To encrypt any data **t** it is exponentiated with the secret **e**: $s = t^e[m]$. Decryption is done by exponentiating the encrypted data with the public part **d**: $t = s^d[m]$ An additional feature of RSA is called **signing**. To **sign** any data set the private key is used. Consequently **verifying** the resulting signature is possible with the public key. Anybody who knows the public is then able to verify this signature. Hence this feature unsuitable to ensure that the data cannot be read by a third party. But for example it is useful for identifying its origin.

4.2 Distributed RSA key generation

For asymmetric cryptography to be of any use in a distributed environment like a P2P network it is also necessary to think of a distributed way to generate the key pair (P, S). It is important that no peer in the network knows the secret key S entirely. So the peers have to jointly generate **shared** RSA keys. Fortunately efficient algorithms to do this already exist like the one proposed by [1].

4.3 Distributed signing and encryption using RSA

RSA has a crucial property: It is a homomorphism. Without this property it would not be possible to use it in a distributed environment. Using the distributed key generation algorithm introduced above all parties know only a part e_i of the secret e after the key pair is generated. However due to the homomorphic property of RSA it is possible to assemble partial signatures t^{e_i} into the full signature t^e. Equation 1 illustrates how the homomorphic property of RSA can be utilised. By applying basic rules of exponentiation it can be shown that instead of computing $t^e[m]$ it is also possible to compute $t^{e_i}[m]$ for every part e_i of the secret e and in the end multiplying the results. In other words it is possible to generate a signature without actually knowing the secret key S as long as all partial signatures are available.

$$t^e[m] = t^{\sum_{i=1}^{n} e_i}[m] = (\prod_{i=1}^{n} t^{e_i}[m])[m] \quad (1)$$

In fact it is not necessary to combine *all* partial signatures. It is sufficient to settle for a ratio or *threshold* **r** on key generation. To retrieve the full signature it is then enough to combine any **r**% partial signatures. This method is based on [5] and [2] and is accordingly called *threshold cryptography*. This feature can be used in a P2P network to sign data, or create certificates for that matter, without any party in the network actually knowing the secret key $S = (e, m)$.

5. PUBLIC KEY INFRASTRUCTURE

In a *Public Key Infrastructure* (PKI) certificates are issued by a *Certification Authority* (CA). A certificate can be used by the party that it was issued to, to authenticate itself by showing it to another party. Such a certificate is signed and issued by the CA. Hence if one party trusts the CA it can also trust the certificate that was issued to the other party. Today the PKI is often used in combination with HTTPS and SSL/TLS for secure interaction on websites with sensitive content. For example on-line-banking or web mail services take advantage of the PKI to authenticate themselves to the user to prevent fraud.

5.1 Certification Authorities

CAs issue digital certificates signed with their private key. If two parties, A and B, do not trust each other but one of them owns a certificate signed by a CA, this party can authenticate itself by showing this certificate. The other party can validate the certification by checking the signature with the CA's public key. Unless both parties have a certificate, though, this authentication can only go one-way.

Of course this only works in the first place if the CA is trusted by both parties. This is also called a *chain of trust* and the CA is often referred to as the *trusted third party* (TTP).

5.2 Disadvantages for P2P systems

In (*pure*) P2P networks there are no central instances. Every peer is treated equally. Consequently an institution like a CA contradicts this paradigm. To fully integrate the concept of a PKI into a P2P network this central CA needs to get distributed across all the peers that participate in the network. Illustration 1 opposes the central PKI and the distributed PKI (DPKI). In the following we propose an approach to build such a DPKI for a P2P network.

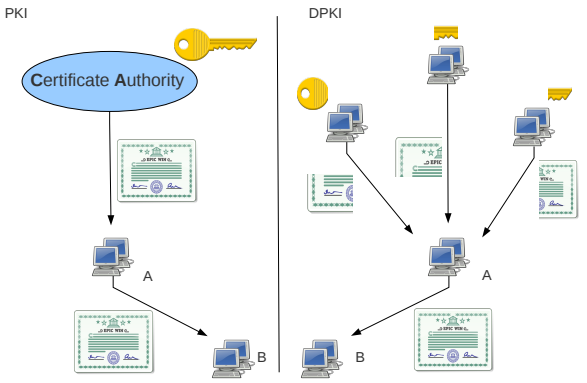


Figure 1: Left: A is issued a certificate by the CA and authenticates itself to B. Right: A is issued partial certificates and assembles the full fletched certificate to authenticate itself to B.

6. PROPOSED DISTRIBUTED P2P PKI

First we take a look at how the distributed PKI is bootstrapped. This includes initial key distribution and the basic layout of the network. Furthermore we explain how the PKI can be maintained by defining a set of operations on a structured P2P network like *Kademlia* as proposed by [3]. In theory, however, all operations discussed in the following can also be implemented for unstructured P2P networks with little extra work [3].

The idea behind this approach is that every node in the network knows only part of the private key $S = (e, m)$ that is used for certification. However, in the beginning this secret has to be generated. It is important that also in this generation phase no peer knows e completely at any time. To achieve this an efficient distributed algorithm already exists for RSA as proposed by [1]. After the generation of the private key S any peer in the network only knows the modulo **m** and $e_i | i \in \{1...n\}$ where n is the number of unique key parts or *shares*. Here it makes sense to introduce the concept of *Sharing Groups*. A Sharing Group consists of nodes that share the same information: Any peer in the Sharing Group i (SG_i) knows only the share e_i. This aggregation of peers into groups has the advantage that e_i is not instantly lost if a peer that knows this part of the secret leaves the network. Every share has a unique *shareId*. At the same time every peer in the network is assigned a unique *peerId*. The *shareId* is a prefix within the peerId in the form $peerId = shareId*$. Consequently when given a *shareId* it can be instantly determined if a peer knows this share (or at least part of it) by looking at its *peerId*. This feature combined with a structured P2P network makes it easy to find required shares that are needed to complete a certification process. Our approach for distributed certification is presented in the following.

6.1 Certification scheme

To perform an actual certification all Sharing Groups have to collaborate. Only if at least one peer of every Sharing Group takes part in the certification process it is possible to generate a valid certificate. An example certification process can look like this:
We assume there are three Sharing Groups ($SG_{shareId}$) in the network. Those are SG_0, SG_{10} and SG_{11}. An exemplary walkthrough is illustrated in 2 and the stages will be explained one by one in the following.

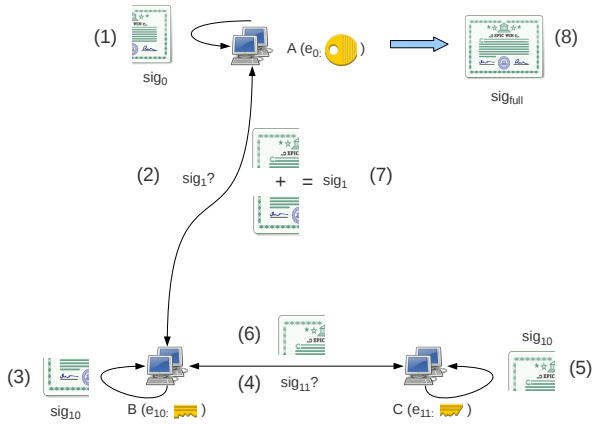


Figure 2: The distributed certification process.

The peer A, which is part of Sharing Group 0 (SG_0), wants to sign a certificate **cert**. Since A knows the share e_0 it can compute the partial signature sig_0 for the certificate **(1)**. The calculation of the partial signature looks like this:

$$sig_0 = cert^{e_0} \tag{2}$$

(Note: Generally when creating a signature a one-way hash function like SHA-1 is applied on the data and the resulting hash is signed. In our example, though, this step is omitted to keep it simple.)

As explained previously when using RSA the full signature can be calculated by multiplying all partial signatures. To generate the full signature $sig_{full} = sig_0 * sig_{10} * sig_{11}$ A requires the help of the other two Sharing Groups. A now asks a peer B that is in SG_{10} for the partial signature sig_1 **(2)**. This also requires A to send the certificate to B. B on the other hand does not know sig_1 completely but only sig_{10} **(3)**. Fortunately it holds that $sig_1 = sig_{10} * sig_{11}$ so B can

ask **(4)** any peer C in SG_{11} for the partial signature sig_{11} **(5),(6)** to calculate sig_1. After B sent sig_1 to A **(7)** the certification process can finish. A only needs to calculate: $sig_{full} = sig_0 * sig_1$. By using this signature A can complete the certification process **(8)**.

6.2 Maintenance operations

Peer-to-peer networks can be very dynamic. This means that peers frequently leave or join the network. Hence it is essential that the partial secrets distributed across the peers are redundantly available. Consequently there are multiple peers that know the same share of the public key S. Those peers are put into *Sharing Groups*. As long as there is at least one peer in each Sharing Group the network's public key S is safe. Before the network can be set up, however, it is necessary to settle for a maximum (max_{SG}) and minimum number (min_{SG}) of members in a single Sharing Group. As discussed above to successfully sign a certificate at least one peer of every Sharing Groups has to participate in the process. This implies that the ratio r of required peers depends on the maximum and minimum size of the Sharing Groups. In fact it holds that:

$$\frac{1}{max_{SG}} < r < \frac{1}{min_{SG}} \tag{3}$$

However, if all peers of a Single Sharing Group leave, the network's private key S is lost and cannot be recovered because the other peers, by definition, do not know S completely. To counter this a few maintenance operations are proposed in the following sections.

6.2.1 Split

If the size of a Sharing Group exceeds max_{SG} peers it should *split*. Without a split the ratio **r** would no longer be within its boundaries (see equation 3). This, however, is not the only reason why keeping the number of Sharing Groups high is a good idea. It also minimises the probability of a certain attack that can occur when there are only a few Sharing Groups. A detailed explanation of the attack is discussed later in section 7.

When a split occurs in a Sharing Group SG_i, the secret all peers in this group know (e_i) is split into two parts: $e_{i0} = \Delta$ and $e_{i1} = e_i - \Delta$ with Δ being a random number. Respectively Peers split into the new Sharing Groups SG_{i0} and SG_{i1}. A peer can determine by itself in which Sharing Group it splits because $peerId = shareId* = i*$ must still hold for i1 as well as i0. Consequently the peerId of a peer determines the Sharing Group it will belong to after a split. The split operation is illustrated in figure 3.

6.2.2 Refresh

The split operation has some drawbacks: Both Sharing Groups that result from a split know the original share e_i. This means that both Sharing Groups can calculate the other Group's share. For example SG_{i0} can calculate $e_{i1} = e_i - e_{i0}$. That is obviously a security flaw because no Sharing Group should know the share of any other Sharing Group.

To achieve this, [3] propose the *refresh* operation: After a split the Sharing Group selects a random other Sharing Group and they swap share information. By doing so the resulting share is no longer calculable by the other peers that split into the opposite Sharing Group. The refresh process is outlined in figure 4.

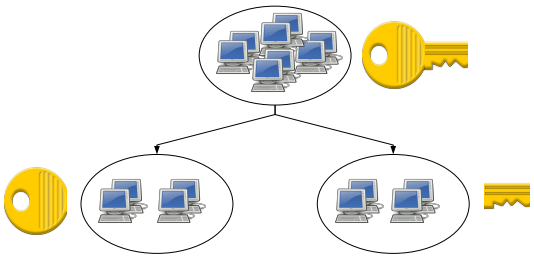


Figure 3: The split operation divides the secret across the new Sharing Groups. Key parts are representing the secrets.

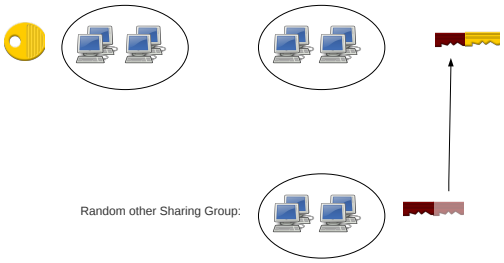


Figure 4: A Sharing Group performing the refresh operation after a split.

6.2.3 Merge

To ensure that the secret $S = (e, m)$ is not lost, it is important that no Sharing Group dissolves because all of its peers disconnect. Hence if the ratio of peers inside a Sharing Group SG_{i0} drops below $\frac{1}{max_{SG}}$ it simply merges with another Sharing Group SG_{i1}. After the merge, the Sharing Group SG_i consisting of all the peers that were previously part of SG_{i0} or SG_{i1} is created. The share all the peers in SG_i know is then simply calculated like this: $e_i = e_{i0} + e_{i1}$ The operation is illustrated in figure 5.

6.2.4 The Sharing Trees

The above operations work well in theory. However, they rely heavily on byzantine agreements. For instance peers always need to monitor their Sharing Group and all of them have to agree to split at some point. The same problem arises for the merge operation, of course. Also when the refresh operation is executed all peers of a Sharing Group

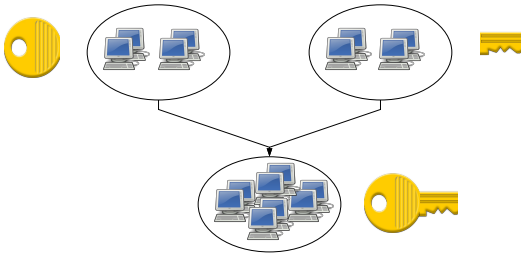


Figure 5: Two Sharing Groups merging together.

have to agree on a random value Δ and a random other Sharing Group. Those agreements are infeasible to fulfil in a practical implementation.

To solve this issue [3] propose the use of *Sharing trees*. Every Sharing Group is associated with such a Sharing tree. It defines for instance how the secret e is divided if a split occurs. This makes it possible for peers to split at different times because they will all split into the same Sharing Group and calculate their new secret in the same way. The secret e can be seen as the root of the main Sharing tree as illustrated in figure 6. If a peer decides to perform a refresh operation

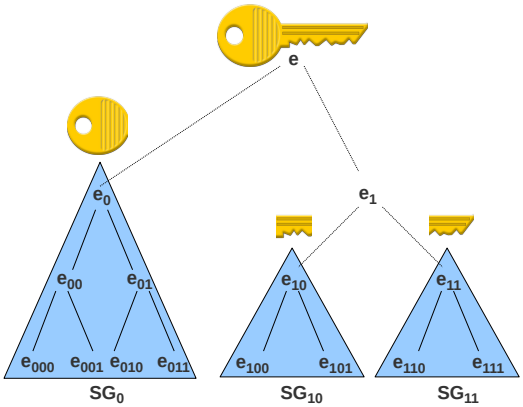


Figure 6: Here three sharing trees are highlighted of the three Sharing Groups SG_0, SG_{10} and SG_{11}.

it can select a random other Sharing Group and the random value Δ that will be exchanged. Then it tells all members of its Sharing Group to perform the refresh with its chosen variables. All peers in both Sharing Groups are able to update their Sharing tree correctly.

7. VULNERABILITIES AND DEFENCE

Just like with any system this proposed approach for a distributed PKI is not perfect. Two straightforward attack vectors are presented in the following along with probabilities of success and defensive measures.

7.1 Attacker in each Sharing Group

The first obvious issue to discuss here is: What happens if collaborating attackers manage to infiltrate every Sharing Group. This scenario would require **n** attackers sitting in **n** different Sharing Groups where **n** is the current number of Sharing Groups.

In this case the attackers can combine their knowledge and assemble the network secret $S = (e, m)$ resulting in a compromised PKI. Generally it can be said that the more Sharing Groups there are in the network the less probable this attack becomes if newly joining peers are always assigned to random Sharing Groups. Say the probability that a peer is an attacker is k_a. Now the amount of members of a Sharing Group i is g_i. The probability that all members of a Sharing Group i are *not* attackers is $(1 - k_a)^{g_i}$. Consequently the probability that there is in fact at least one attacker in SG_i is $1 - (1 - k_a)^{g_i} < 1$. Now we can also calculate the probability that there is *at least one* attacker in each Sharing Group:

$$P_{att} = \prod_{i=1}^{\#SG} 1 - (1 - k_a)^{g_i} \quad (4)$$

This equation proves that the probability of this attack (P_{att}) decreases if the number of Sharing Groups (#SG) increases. This is the most important reason why the *split* operation exists. By applying this operation it is possible to ensure that a reasonable high number of Sharing Groups exist. A visual representation of such an attack can be seen in figure 7.

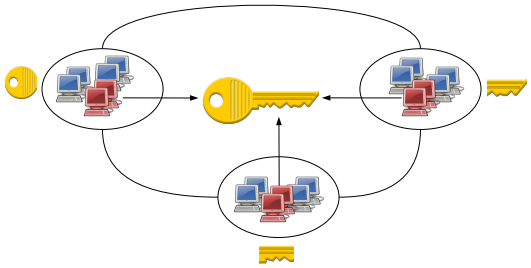


Figure 7: In this illustration three attackers from three different Sharing Groups participate in the retrieval of the secret key.

7.2 Misbehaving Sharing Group

Another possible attack consists of attackers "taking over" a single Sharing Group. This attack does not expose the network's private key S like the one discussed previously, but if a certificate needs to be signed the compromised Sharing Group can block the process by not answering to any requests. This attack, if successful, also makes the PKI unusable. Figure 8 illustrates the attack.

We can now look at the probability of this attack: Let k_a be again the probability that a peer is evil and g_i is the number of peers in Sharing Group i. The probability that

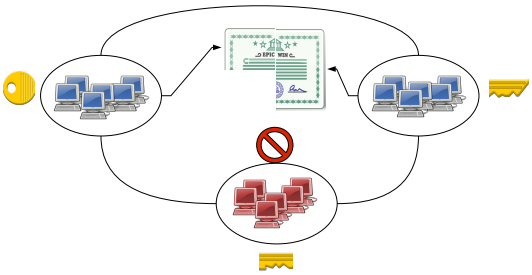


Figure 8: In this illustration two of three Sharing Groups participate in the certification process. The Group on the bottom, however, denies the final part.

there are only attackers in SG i is: ${k_a}^{g_i}$. Accordingly the probability that there is at least one peer in the SG that is not an attacker is: $1 - {k_a}^{g_i}$. Consequently the probability that there is at least one Sharing Group containing only attackers is:

$$P_{att} = 1 - \prod_{i=1}^{\#SG} 1 - {k_a}^{g_i} \tag{5}$$

Paradoxically every *split* operation theoretically raises the probability of this attack, because the less peers there are in a Sharing Group the more likely it is that all of those peers are attackers.

In conclusion it is obvious that the size limits of the Sharing Groups determine the success probabilities of those two attacks. Unfortunately minimising the likelihood of success of one attack increases the probability of success for the other. In other words the Sharing Group size describes a trade-off between those two issues and should be chosen wisely.

8. CONCLUSION

We presented an approach that can be used to distribute a public key infrastructure in a P2P network. It allows to put away with a central Certification Authority by distributing the certification process. Future work can be focused on making the system more robust against attacks. Also, as with any PKI implementation, certificate revocation is still an issue. The creation and distribution of the revocation lists (CRLs) would have to be distributed just like the decision process that determines what certificates are to be revoked and for what reason. An advantage in terms of security and availability of CRLs is that denial of service attacks against the PKI could be avoided. If the CRLs are properly distributed across a large number of peers the probability that a certification process fails because of an unavailable CRL could be minimised.

9. REFERENCES

[1] D. Boneh and M. Franklin. Efficient generation of shared rsa keys. *J. ACM*, 48(4):702–722, 2001.

[2] Y. Desmedt and R. Holloway. Some recent research aspects of threshold cryptography. In *In Proc. of the 1st Intl. Information Security Workshop*, pages 158–173. Springer-Verlag, 1997.

[3] F. Lesueur, L. Mé, and V. V. T. Tong. An efficient distributed pki for structured p2p networks. In H. Schulzrinne, K. Aberer, and A. Datta, editors, *Peer-to-Peer Computing*, pages 1–10. IEEE, 2009.

[4] R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *ASIACRYPT '01: Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security*, pages 552–565, London, UK, 2001. Springer-Verlag.

[5] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[6] L. Zhou, F. B. Schneider, and R. V. Renesse. Coca: A secure distributed online certification authority. *ACM Transactions on Computer Systems*, 20:329–368.

Weaknesses of today's Internet architecture

Felix Schindler
Betreuer: Nils Kammenhuber
Seminar Innovative Internettechnologien und Mobilkommunikation SS2010
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: schindlf@in.tum.de

ABSTRACT
This paper describes the main weaknesses of today's Internet architecture, and the problems that arise from these weaknesses. In addition to that it gives a short overview over the assumptions on which the original architectural decisions were based, and why they do not hold anymore. There are some serious design flaws that really keep the Internet from showing its full potential when it comes to performance, and also in terms of usability. These weaknesses pose a big threat to the future of the Internet. Although there is a lot of research on future Internet technologies going on at the moment, still a lot of effort will be needed to overcome these barriers.

Keywords
Internet, Architecture

1. INTRODUCTION
The history of the Internet is a story of success and growth, and at first sight it seems to make a good job. But if you take a closer look behind the scenes, you start to realize that this is not the case. This paper gives an overview over the design decisions on which today's Internet is based. It also tries to give some insight on why the architecture is struggling to meet today's demands and the demands of the future. All this is becoming more and more important, as it has a great influence on how the Internet will look like in the future.
The first part of this paper focuses on problems that are based on the simple core functionality of the Internet. In the second part we will show why the unexpected growth in itself is causing more and more trouble. After that there will be a section about the most common security issues and why their importance is growing. Later we show why the change from a relatively static to a completely dynamic network is causing ISPs some serious headache. Last, we will look at problems that are based on combinations of the points described before.

2. SMART EGDE, DUMB CORE
For a better understanding of this paper it should be clear that when we talk about the Internet, the core Internet or the core network in almost all cases we talk about IP and the network layer and not about transport layer protocols.
To get a better understanding of the architecture of today's Internet and the resulting problems, a few points about the basic design decisions [1] have to be made clear first. The core architecture of the Internet was planned and implemented to be simple and robust. It offers a best effort service trying to forward small packages of data (IP packets [3]) to a certain destination. Apart from that, next to no additional services are offered. If you want to send data to another host in the Internet the data is split into small packets, containing a source address and a destination address. Then it is sent into the network. Within the network the nodes (routers) that receive those packets try to determine the best route to the destination of the packets, and just forward them into this direction.
What is really characteristic about the Internet is that the source of a packet can never be sure that the packet it sent actually arrived at the destination. If the packet got lost on the way for whatever reason, the sender is not informed in most cases. In short terms, the Internet is a best effort packet switching network [3].
In general it was tried to keep the components that make up the core Internet (see Figure 1) as simple as possible. The result was a stateless network. Routers do not store or use any state when they forward packets through the network, except for the routing information. One reason for that certainly was the hardware performance during the first years of the Internet. Keeping state for every connection would have overwhelmed the capabilities of routers, or at least made them much more expensive. In addition to that, the stateless approach has a considerable advantage. If routers kept state and then failed for whatever reason, all state were lost and had to be reestablished after recovery. This would result in dropped connections and lost data, which really is not desirable.

As we have seen, the services offered by the core Internet are very limited on purpose. This lead to the fact that whenever additional functionality apart from simple packet forwarding was needed, it was implemented not at the network layer but at higher protocol layers. This course of action has been practiced up until today. It has become common sense that new functions should be introduced at higher levels in the network protocol stack, if they do not add considerable advantages when implemented at lower layers. Therefore, the core network architecture and protocols hardly changed over the years after their introduction [2]. Another reason for the reluctance to to introduce new functions is the size of the Internet. It is really difficult to change big networks, as it would require to apply changes to a huge number of systems all over the world. Especially for network equipment like routers, today this is nearly impossible without threatening

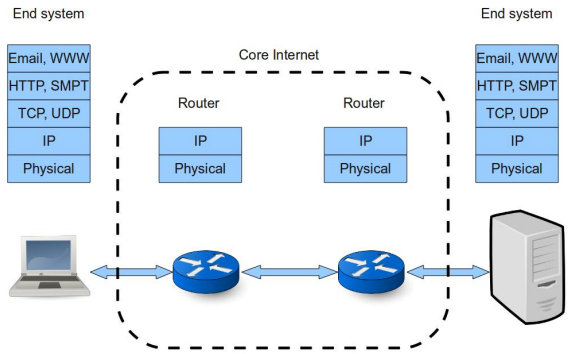


Figure 1: Protocols used in the Internet

the functionality of big parts of the Internet.

2.1 Lack of Quality of Service

The number of applications and protocols used in the Internet is growing, and with that the number of applications that rely on a certain performance provided by the underlying network connection. These applications and protocols require a certain Quality of Service (QoS) [16] in order to work properly. Applications like IP telephony or online gaming need considerable bandwidth or low latency in order to provide a good user experience. Therefore it would be desirable to notify the network about these demands, to reserve resources along the path of communication within the network to meet these demands. This also includes the separation of traffic into different classes which then can be given different priorities. For instance it would be preferable to give Voice over IP (VoIP) traffic a higher priority contrary to a simple file download. For the duration of a VoIP call, the network should have the ability to throttle the speed of the file download, so as to be able to offer enough resources to keep up the voice quality and the low latency.

The current architecture and the current protocols do not offer this kind of functionality, because it contradicts the principle of keeping state out of routers. For every connection or data flow, every router on the communication path would have to store some information about how to handle the incoming data. Not only is this an additional burden for the router in terms of memory consumption and processing time; it also introduces new problems that arise when a route suddenly changes. This may be the case when new routes are introduced or old ones are taken down. Another question is how the reservation of resources is propagated through the network along the communication path.

Of course there are protocols which offer resource reservation in the network [4], and even IP offers a header field for QoS, but hardly anybody uses them. Some providers use QoS mechanisms within their networks to guarantee certain services to single customers, but QoS across the borders of ISPs is just not available.

Instead of using QoS, IPSs try to provide the needed resources for certain protocols by using fast and expensive network equipment. This approach is called over provisioning [11]. Providers need to come to an agreement including a standard protocol, a proper billing mechanism and a way to guarantee QoS across the borders of single ISPs. Otherwise, over provisioning will stay the method of choice, despite being costly.

2.2 No congestion control in the network

One thing which is really remarkable is that the core Internet usually does not offer congestion control. Of course there are mechanisms which implement congestion control at the network layer, but they are not compatible to each other. The result is that they are used only locally and in most cases they are not used at all. The most widespread form of congestion control is implemented at the transport layer, in end systems in the Transmission Control Protocol (TCP) [6] [9] and similar transport layer protocols [7] [8]. And even there it was introduced later in the mid 1980s after the Internet collapsed a few times due to congestion [2]. It was a quick fix intended to face an imminent problem, and was thought to be a temporary answer to the congestion control problem [2]. An indicator for this is the fact that other transport layer protocols which are commonly used do not offer congestion control at all. This is a cause for concern, as more and more data intensive applications like audio streaming or video streaming refrain from using TCP. They prefer the User Datagram Protocol (UDP) [13] as transport protocol because of its high throughput, low overhead, and low latency. This leads to two problems. First the applications using UDP as transport protocol violate the principle of fairness. Fairness in this case means that all applications should obtain an equal share of the available bandwidth. TCP provides this kind of fairness while UDP does not [9] [12]. The second problem that may arise is even more severe. Due to the fact that there is no congestion control built into UDP, it is possible to overwhelm the network with traffic and therefore causing it to collapse [9].

Congestion control in end system at the transport layer might not be the best solution, because end systems usually have only little information about what is going on in the network [15]. The core of the network, which has all the information needed for proper congestion control, should be able to make much more sensible decisions when it comes to congestion control. This might also lead to a more effective usage of the bandwidth of links connecting parts of the network.

Instead of implementing congestion control at the network layer, the main strategy of ISPs to overcome this problem is again to use over provisioning. As mentioned before, this might not be the best solution although it worked reasonably well in the past. But in the long run it might prove as insufficient [11].

2.3 No traffic filtering in the network

In today's Internet there is no possibility to unsubscribe from unwanted traffic. All traffic you do not want to receive could be filtered within the network, before it reached your end systems. As we will see this approach has major advantages.

Unfortunately all traffic sent to you will reach you in the end. If you do not want it you have to filter it yourself. This is a serious drawback. It means that unwanted traffic has to cross the Internet before it can be discarded at the destination. This traffic thereby uses a lot of valuable resources like bandwidth and processing time in routers, just to be thrown

away in the end. Spam, for example, causes a considerable amount of today's Internet traffic, and it would be a big advancement to be able to filter this kind of traffic in the network as soon as possible. This course of action makes even more sense if you consider DoS attacks [14]. These attacks try to exhaust the victim's network and server resources by overwhelming them with huge amounts of traffic. This attack is really hard to protect against, unless you are able to intercept the traffic in the network, before it reaches the victim.
Being able to unsubscribe from unwanted traffic, and being able to block unwanted traffic in the network might even be sensible from a financial point of view. It would drastically reduce the amount of money which is spent by ISPs and companies to protect from DoS-attacks, spam and other unwanted traffic that poses a threat to their systems. It also might add to the security as there are several independent filtering points throughout the whole Internet and not just one at the end system of the receiver which poses at potential single point of failure.

2.4 No Multicast

Another technology that is still lacks widespread deployment in the Internet is multicast[17]. In contrast to unicast addressing, more than one host can be reached by sending to a single address. Host which are reachable with the same multicast address form a multicast group. When sending data to a multicast address, it is distributed to all members of this multicast group. This scenario offers several advantages over the common unicast approach. A server that distributes content does not have to be aware of all the recipients that actually want the data. The server just sends the content once to the multicast address, and the network takes care of distributing the data to all members of the multicast group. This introduces some big improvements for the server, because it does not have to handle connections to all clients that want content. In the best case scenario, the server has to send the data only once, and is still able to reach thousands of clients. The multicast approach also reduces bandwidth consumption within the network, as the data that is sent is only duplicated at routers that have more than one multicast group member attached to them (in network duplication, see figure 3). When using unicast data has to be sent n times for n clients (source duplication, see figure 2).

A good example for an application which could make use of this technology extensively is Internet Television. With thousands of people watching the same program at the same time, multicast would drastically reduce the load on the servers and the network. Another scenario which can be considered, would improve the propagation of updates. Instead of thousands of clients that periodically contact the server to ask for new data, they join a multicast group. Once there is an update at the server, it is distributed to the multicast address.
When looking at these advantages one might wonder why the use of multicast is not common in the Internet. Let us take a look at reasons. First there are a lot of different multicast routing protocols which complicate the task of building a distributed multicast architecture. In addition to that, some multicast routing protocols are limited to intra-AS use. Therefore it is hard, or even impossible to build

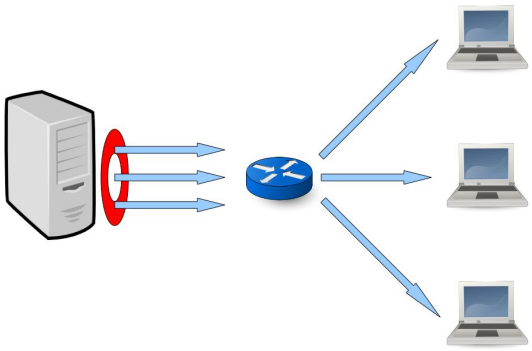


Figure 2: Source duplication

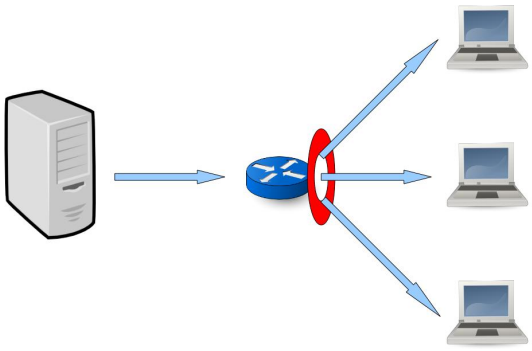


Figure 3: In network duplication

multicast structures which overcome the borders of single ASes [18].
Another reason for the slow deployment of muticast is that, at least at the moment, the management overhead is still too big. This means that it is more cost effective to stick to unicast, despite its higher bandwidth consumption, than to effectively manage multicast routing [18].
Similar to QoS, the lack of a working billing infrastructure keeps multicast from being deployed on a large scale.

3. INTERNET SIZE

Let us now take a look at the size and the growth of the Internet, which is becoming more and more of a problem. During the development phase it was considered to be a network of intermediate size, connecting a few hundred, maybe few thousand military, scientific and educational institutes all over the world. The most important decision that is based on this consideration is the size of the IP address space. IP addresses consists of 32 bits which leads to roughly 4 billion possible host connected to the Internet. As will we show later on, IPv6 (see section **??**) offers bigger addresses. For quite a while this seemed to be more than enough. But then the Internet started to grow exponentially (see figure 4), and

people realized that we would eventually run out of IP addresses.

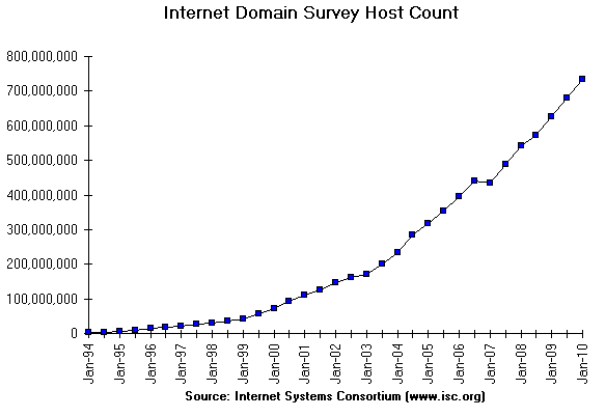


Figure 4: Exponential growth of the Internet [34]

Calculations have shown that the last unused IP addresses will be given to Regional Internet Registries (RIRs) in 2011, and that the RIRs themselves will give away their last unused addresses in 2012. Although there are possibilities to cope with that problem at least for a certain period of time, this is one of the biggest and most imminent threats to the Internet.

3.1 Network Address Translation

The first thing which has to be pointed at when talking about Network Address Translation (NAT)[20] is that it was introduced to provide a temporary solution for the address space depletion. But as time went by, NAT became more and more common and started to become a problem itself.
NAT tries to cope with the depletion of IP addresses by hiding whole networks behind one public address. This is done by mapping the local IP address and the port to the public IP address and a public port. At first sight this seems like a sensible solution, but if you take a closer look, NAT introduces a whole new set of problems.
Although devices behind a NAT are able to connect to hosts in the Internet, connections in the opposite direction usually fail. This breaks nearly all applications that rely on end-to-end connectivity. Famous examples are FTP and peer-to-peer applications. Apart from breaking end-to-end connectivity, it also introduces problems for programs that send their own address to other hosts. When someone tries to to establish a connection using SIP [21], this will not work because his own address which he sent is from the private network, and not reachable from the Internet.
Another big problem when it comes to Network Address Translation is that there is no common standard of how a NAT device should be implemented. This is really painful for developers of network applications, as they have to provide solutions for all different kinds of NATs. This has lead to several different frameworks [32] that try to help applications to determine the kind of NAT which is used, and offer functionality to overcome the NAT traversal problem.
Apart from not being standardized, NAT usually blocks all transport layer protocols that that are neither TCP nor UDP and therefore prevents new transport layer protocols [7] [8] from being propagated.

The biggest problem introduced by NAT is its extensive and widespread use. NAT has become so common that some people claim that the Internet has changed from a network connecting networks to network connecting NATworks.

3.2 IPv6 deployment

IPv6 [5] is an advanced version of the current network layer protocol IPv4. It was developed to take care of the problems of IPv4. First IPv6 offers 128 bit addresses that allow about $3.4x10^{38}$ hosts, which addresses the problem of IP-address depletion. It uses fixed length headers which reduces processing efforts within routers. Another feature that was introduced to reduce the processing load on routers is that there is no possibility to fragment IP packets within the network. IPv6 also introduces new features for a better support of QoS and multicast. Even network layer security is provided by IPv6.
As you can see IPv6 offers a lot of improvements over IPv4, and it officially is the successor of IPv4. The problem is that it has been like this for over 10 years. Since its definition in 1998, people thought that the switch to IPv6 would be just around the corner. Nowadays it is supported by protocols needed to run the Internet (DNS, routing protocols, etc.) and by all operating systems running on end systems. But still IPv6 traffic accounts for a very small percentage of the overall Internet traffic (see figure 5). If you take a look at the top one million websites of the world, only 0.15 percent offer an IPv6 website [22].

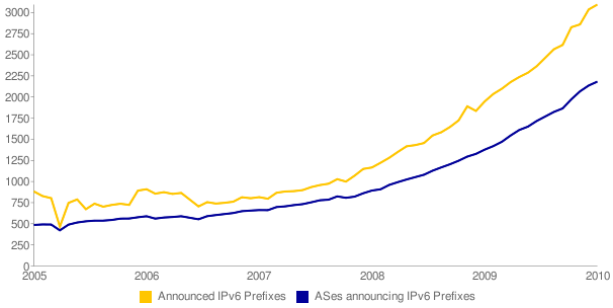


Figure 5: IPv6 Deployment in the Internet [26]

Why are people so reluctant to switch to something that is obviously better than the current system? There are several reasons for that. First is the question of how the transition should be made. There are two conflicting scenarios. The first one introduces a flag day, on which the whole Internet replaces IPv4 by IPv6. By looking at all the complex systems involved making the Internet work, you will realize that this approach just would not work. It would take days if not weeks to get the basic functions operating and even more important, interacting again. This would be a global disaster as whole economies are based on and rely on the Internet. Another strategy is a slow transition from IPv4 to IPv6 with both protocols coexisting for a certain period of time. This includes tunneling IPv6 traffic over IPv4 networks, and a so called dual stack [23]. This dual stack provides IPv4 and IPv6 capability to the TCP/IP protocol stack at the same time.
Another problem is that at the moment there is no financial gain in switching to IPv6. Actually, quite the opposite is true. The switch requires a lot of work and financial effort.

This poses a chicken and egg problem. In order to use IPv6 at end systems it has to be provided by content providers first. Content providers cannot switch to IPv6 if ISPs do not offer it. And the ISPs do not offer it because end users do not ask for it, because there is content distributed over IPv6.
As you can see, the switch to IPv6 is taking place slowly, and both protocol versions will coexist for quite a while and it is still not clear when IPv6 will be deployed on a large scale. But in the long run it will replace IPv4 as the problems it addresses are becoming more and more serious in the future.
The switch to IPv6 is a good example of the attitude towards fundamental changes in the architecture of the Internet: "Never touch a running system." At least if it is not absolutely necessary. This shows how inflexible the Internet has become over time in spite of its modular architecture. This inflexibility is sometimes called ossification [2]. It also shows how dependent the world has become on a working Internet.

4. DYNAMIC NETWORK

As we have seen before, the Internet has grown exponentially in the past and will keep on doing so. With the size and the widespread use a new problem arises. The Internet is changing constantly. New networks join, old networks leave or move somewhere else. New connections are introduced, old ones are discarded. This fact is putting more and more pressure on the design which was based on the assumption that the Internet would be a relatively static network.

4.1 Slow routing convergence

The Internet consists of more or less independent networks that are interconnected. These so called Autonomous Systems (ASes) share routing information using the Border Gateway Protocol (BGP) [24] in order to provide a globally consistent information basis on which routing decisions can be made. Apart from the routing tables which have become huge (> 300.000 entries, see figure 6) [25], BGP has difficulties coping with the constant change happening in the Internet. The problem is that it takes time to distribute updated routing information throughout the Internet to reach a new consistent state. Not only is the time the updates take a problem in itself, but during that time the global routing information is inconsistent. If some parts of the network make routing decisions based on new information, and some parts on old information, they might make incompatible decisions. This can lead to data being routed to dead ends, where it is dropped and therefore lost. Established connections break, and even whole parts of the Internet might be unreachable until a consistent state is reestablished. Due to the fact that the routing information basis is growing constantly, and therefore changing constantly this is a serious problem, which is hard to come by. Although these glitches are usually only temporary, they have to be approached nevertheless, as reliability is asked for by normal end users, and needed by most companies. In addition to the annoyance these glitches cause there is also the risk of substantial financial loss. If safety critical systems are involved failures due to slow routing convergence might even endanger people.

4.2 No locator/ID split

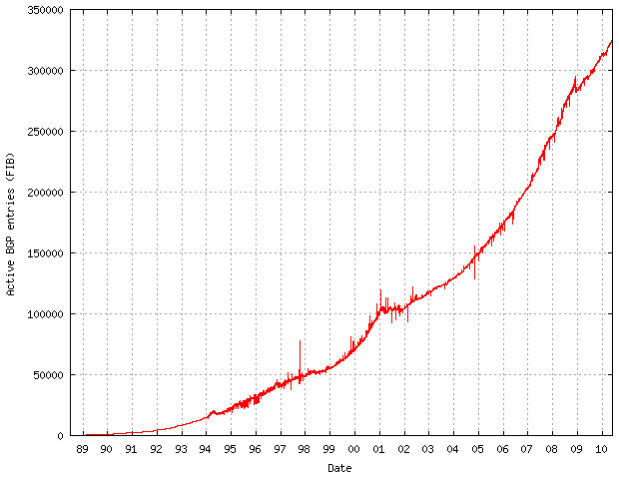


Figure 6: Growth of BGP routing tables [25]

If we look at an IP address we realize that it fulfills two functions. First it points to a certain location within the network which is needed for proper routing. Second it provides a unique identifier for a device. When you think of stationary devices this does not seem like a problem. And in most cases it is not. But change also affects the location of devices. With the spreading of mobile devices like laptops, handhelds and even mobile phones, end systems tend to move to different locations in the network on a regular basis. This means that whenever they change their location they will get assigned a new IP address, and the result is that they cannot be reached anymore using the old address. So whenever a device moves somewhere else, it has to inform all communication partners about its new IP address. Otherwise they will not be able to contact that device any more. In addition to that, established TCP connection are dropped when the IP-address of one of the communication partners changes.
Apart from that, even a lot of stationary devices change their IP address on a regular basis. They either get IP addresses assigned dynamically by their home network (e.g. DHCP), or their access providers give them a new address whenever they connect to the Internet (most dial up connections work that way).
With the rise of mobile systems, a separation of the location and the device identifier would be preferable. This is called the locator/ID split. It basically decouples the location within the network from its globally unique identifier. When the device moves only the locator changes, but it still can be reached using its unique identifier.
This approach offers advantages for different scenarios [27]. First it becomes easier to migrate whole networks from one location to another, without having to renumber all hosts within this network. The second point is that it is less complicated for networks to be attached to the Internet at two different points. Company networks are often connected via two different ISPs to have redundancy in case of one ISP having problems. Third, the locator/ID split can reduce the size of routing tables, because the network locators could be aggregated much more efficient than today's IP addresses [28]. Systems that address this problem are the Domain Name System (DNS) which basically provides a service to resolve

names into IP addresses. The problem with DNS is that changes are propagated too slowly to provide a reliable solution for the locator/ID split. Another discussed solution divides IP addresses into a dynamic locator and a static identifier. The question is whether to include this functionality into the core network, or to build another system on top to provide it.

5. LACK OF SECURITY

During the first years of its existence, the Internet was a network connecting mostly academic, military and scientific institutions. People who used it were scientists and technicians. It was a situation where people using the Internet could be trusted, because at that time nobody would try to harm other users or the infrastructure itself [29]. First, everybody was knowledgeable enough not to break something by accident. Second, the Internet was an environment where people cooperated to achieve certain goals. Most important is the fact that at that time, no financial profit could be made by attacking someone or something.
This situation of trust lead to the fact that until today the core of the Internet is lacking basic security features. Security was thought to be an additional "nice to have" specialty. Whenever security was needed, it had to be implemented at the edge of the Internet, in higher layers of the protocol stack.

5.1 Lack of encryption

When you look at the most common applications and protocols used today, like email, WWW or even IP itself, you realize that they offer no encryption whatsoever. For a lot of applications this does not matter, because they actually do not require encrypted traffic. When surfing the Web, in most cases it just does not matter if others are able to know what you are looking at. But more and more people use websites where a login is required to access a certain kind of service. This is problematic, as unencrypted traffic can be read by potential attackers, who are able to gain knowledge of passwords and other login data. When you look at email this is even more severe, because emails often contain personal information or even passwords which are not intended for others.
Another problem when it comes to unencrypted traffic in the Internet is that most users are unaware of the risks or simply ignore them. Sentences like "I have nothing to hide", or "Who would want to attack me¿' are quite common. Although this state of mind is receding more and more, security features are often left unused by end users. The reason for this is that using security functions requires additional knowledge and effort. They often interfere with simplicity and the ease of use of applications.
Of course there are solutions that provide encryption for Internet traffic like SSL/TLS [30] which offers transport layer security or IPSec [31] which actually works on the network layer. But they all have to be used explicitly. As IP is the protocol all traffic uses to traverse the Internet, it is the place where encryption should be introduced and used by default. The network layer is the layer where a common security policy can be enforced, transparent to all protocols and application which are used in the Internet. With the growing need for security, the time has come to introduce encryption as a core feature and not just as a "nice to have" addition. As said before, IPv6 offers network layer security but still lacks a global deployment.

5.2 Lack of cryptographically signed protocols

Similar to the lack of encryption is the lack of digitally signed protocols. They provide two things which encryption alone cannot do. Data integrity is the first one. It protects data from being altered on the way from sender to the receiver without being detected. The second is authentication which identifies the communication partner on a reliable basis. Combined with encryption, these two features make the most important building block of secure communication over the Internet.
Especially when it comes to financial transactions made over the Internet, encryption alone is not enough. In fact, encryption is often pointless if you cannot tell whether your communication partner is who he claims to be. The same goes for data integrity.
Similar to encryption there are protocols that provide these features. And like encryption authentication and data integrity have become important enough to be added as a core functionality to the Internet.

6. LOWEST COMMON DENOMINATOR = HTTP

As we have seen before, the Internet has become a more and more hostile environment in which nobody can be trusted. This lead to the widespread use of firewalls which try to block unwanted and malicious traffic either entering or leaving a network. As described before, NATs also introduce a certain kind of traffic filtering entities. For applications that do not belong to the group of standard Internet applications (Email, WWW, etc.) it is becoming more and more difficult not to get filtered by these devices. Whatever firewall or NAT is used, HTTP traffic is the most likely not to get filtered. This has lead to more and more applications using HTTP to exchange data over the Internet. At first sight, this may not seem too problematic, as HTTP is suitable for a broad range of different applications.

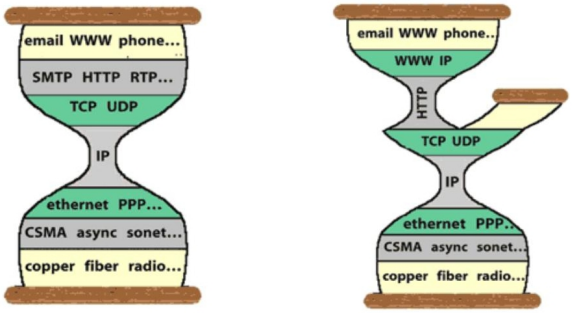


Figure 7: The change of the hourglass model of the Internet [33]

However, this makes it more and more difficult to distinguish between real HTTP traffic and traffic from other applications which can compromise the security provided by firewalls. A good example here is Skype (a widely use VoIP application) as it waits for connections on port 80 and port

443 which are usually used for HTTP and HTTPS connections. In addition to that HTTP uses TCP as transport layer protocol which makes it unsuitable for traffic that requires a constant bit rate like audio or video streams. In figure 7 you can see how the hourglass model of the Internet has changed and HTTP has become sort of the "new" transport protocol.

7. CONCLUSION

Although the Internet seems to be working quite well, we have shown that it is facing problems. Especially the simple core design is starting to show its age, and probably has to be changed in the future, to be able to keep up with the ever growing demand for better performance and functionality.
For almost all problems shown in this paper, there is at least one proposed solution. The main problem is that it becomes more and more difficult to introduce these new ideas on a global scale. The Internet has become too important to be used as a test setup where you can break things. Another point is that there is no common sense about which direction the development should take, as there are different approaches competing against each other. Therefore it is important to switch to IPv6 as soon as possible, as it provides a foundation which deals with at least some of the weaknesses described here.
Finally the time has come where the simple architecture of the core Internet despite its success is starting to show its age. It is necessary to break with the old paradigm of keeping the core simple and introduce new functionality to meet the growing demands for new services.

8. REFERENCES

[1] David D. Clark: *The design philosophy of the DARPA Internet protocols*, Symposium proceedings on Communications architectures and protocols, Stanford, California, United States,Pages: 106-114, 1988

[2] M. Handley: *Why the Internet only just works*, BT Technology Journal 24, 3, July 2006

[3] J. Postel: *Internet Protocol*, RFC 791, IETF, September 1981, `http://tools.ietf.org/html/rfc791`

[4] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin: *Resource ReSerVation Protocol (RSVP)*, RFC 2205, IETF Network Working Group, September 1997 `http://tools.ietf.org/html/rfc2205`

[5] S. Deering, R. Hinden: *Internet Protocol, Version 6 (IPv6)*, RFC 2460, IETF Network Working Group, December 1998, `http://tools.ietf.org/html/rfc2460`

[6] J. Postel: *Transmission Control Protocol*, RFC 793, IETF, September 1981, `http://tools.ietf.org/html/rfc793`

[7] L. Ong, J. Yoakum: *An Introduction to the Stream Control Transmission Protocol (SCTP)*, RFC 3286, IETF Network Working Group, May 2002, `http://tools.ietf.org/html/rfc3286`

[8] E. Kohler, M. Handley, S. Floyd: *Datagram Congestion Control Protocol (DCCP)*, RFC 4340, IETF Network Working Group, March 2006, `http://tools.ietf.org/html/rfc4340`

[9] Sally Floyd, Kevin Fall: *Promoting the Use of End-to-End Congestion Control in the Internet*, IEEE/ACM Transactions on Networking, Volume 7, August 1999

[10] Murat Yuksel, K. K. Ramakrishnan, Shivkumar Kalyanaraman, Joseph D. Houle, Rita Sadhvani: *Value of Supporting Class-of-Service in IP Backbones*,

[11] Yaqing Huang, Roch Guérin: *Does Over-Provisioning Become More or Less Efficient as Networks Grow Larger?*, Dept. of Electrical and Systems Engineering, University of Pennsylvania, 2005

[12] Mohammad A. Talaat, Magdi A. Koutb, Hoda S. Sorour: *A Survey on Unicast Congestion Control Protocols for Media Traffic*, IJCSNS International Journal of Computer Science and Network Security, Vol. 9, No. 3, March 2009

[13] J. Postel: *User Datagram Protocol*, RFC 768, IETF, August 1980, `http://www.faqs.org/rfcs/rfc768.html`

[14] David Moore, Colleen Shannon, Douglas J. Brown, Geoffrey M. Voelker, Stefan Savage: *Inferring Internet denial-of-service activity*, ACM Transactions on Computer Systems (TOCS), Volume 24, Issue 2, May 2006

[15] Lefteris Mamatas, Tobias Harks, Vassilis Tsaoussidis: *Approaches to Congestion Control in Packet Networks*, Journal of Internet Engineering, Vol. 1, No. 1, January 2007

[16] R. Braden, D. Clark: *Integrated Services in the Internet Architecture: an Overview*, RFC 1633, IETF Network Working Group, June 1994, `http://tools.ietf.org/html/rfc1633`

[17] Stephen E. Deering, David R. Cheriton: *Multicast routing in datagram internetworks and extended LANs*, ACM Transactions on Computer Systems (TOCS), Volume 8, Issue 2, May 1990

[18] C Diot, BN Levine, B Lyles, H Kassem, D Balensiefen: *Deployment issues for the IP multicast service and architecture*, IEEE Network, 2000

[19] Ian Brown, Jon Crowcroft, Mark Handley, Brad Cain: *Internet Multicast Tomorrow*, The Internet Protocol Journal, Vol. 5, No. 4, December 2002

[20] Pranitha Anand: *Network Address Translation* University of Alabama, Birmingham

[21] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler: *SIP: Session Initiation Protocol*, RFC 3261, IETF, August 2002, `http://www.ietf.org/rfc/rfc3261.txt`

[22] *Internet addressing: Measuring Deployment of IPv6*, OECD, April 2010

[23] E. Nordmark, R. Gilligan: *Basic Transition Mechanisms for IPv6 Hosts and Routers*, RFC 4213, IETF Network Working Group, October 2005, `http://tools.ietf.org/html/rfc4213`

[24] Y. Rekhter, T. Li, S. Hares: *A Border Gateway Protocol 4 (BGP-4)*, RFC 4271, IETF, January 2006, `http://www.ietf.org/rfc/rfc4271`

[25] *CIDR REPORT for 31 May 2010*, `http://www.cidr-report.org`

[26] `http://www.ipv6actnow.org/info/statistics/`

[27] D. Farinacci, V. Fuller, D. Meyer, D. Lewis:

Locator/ID Separation Protocol (LISP), IETF Network Working Group Internet-Draft, March 2009, http://tools.ietf.org/html/draft-farinacci-lisp-12

[28] Bruno Quoitin, Luigi Iannone, Cédric de Launois, Olivier Bonaventure: *Evaluating the Benefits of the Locator/Identifier Separation*, Proceedings of 2nd ACM/IEEE international workshop on mobility in the evolving Internet architecture, 2007

[29] Randall J. Atkinson: *Security for the Internet Protocol*, Naval Research Laboratory, Wasington, November 1995

[30] T. Dierks, E. Rescorla: *The Transport Layer Security (TLS) Protocol Version 1.2*, RFC 5246, IETF, August 2008, http://tools.ietf.org/html/rfc5246

[31] S. Kent, R. Atkinson: *Security Architecture for the Internet Protocol*, RFC 2401, IETF, November 1998, http://www.ietf.org/rfc/rfc2401.txt

[32] J. Rosenberg, R. Mahy, P. Matthews, D. Wing: *Session Traversal Utilities for NAT (STUN)*, RFC 5389, IETF, October 2008, http://tools.ietf.org/html/rfc5389

[33] Dave Thaler: *Evolution of the IP Model*, IETF Journal, Volume 4 Issue 3, February 2009, http://www.isoc.org/tools/blogs/ietfjournal/?p=454

[34] Internet Systems Consortium: *The ISC Domain Survey*, http://www.isc.org/solutions/survey

The Traceroute zoo

Lukas Schwaighofer
Advisors: Dirk Haage, Johann Schlamp
Seminar course Innovative Internet Technologies and Mobile Communication, SS2010
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: schwaigh@in.tum.de

ABSTRACT
The well known traceroute utility can be used to gather information about the path packets take through a network. Since traceroute is very error-prone, the results may be inaccurate or even unusable. This paper will introduce and discuss some of the different traceroute implementations available.

Keywords
traceroute, load balancers, nmap, traceroute-nanog, Paris traceroute, DisCarte

1. INTRODUCTION
Especially for system administrators, it is very useful to discover the route to a specific destination in order to locate network problems. This task is mostly achieved with the well-known traceroute tool. The Internet Protocol (IP) does not provide any sufficient functionality for route discovery, although two IP features will be discussed in Sections 4.3 and 4.5.1. As illustrated in Figure 1 the only available information is necessarily the first hop of the path. In order to unravel the subsequent hops, the Time To Live (TTL) header field is exploited. While this approach works good enough within small and simple networks, the complexity of the Internet causes inconsistencies. The main reason are non-standard router implementations, firewalls and load balancers [1]. Approaches to deal with these problems and their success will be discussed in this paper.

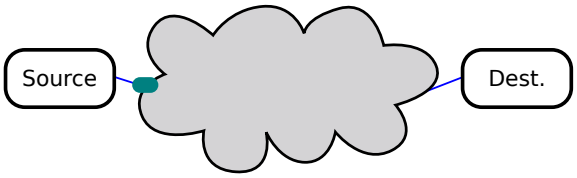


Figure 1: The path to the destination is unknown

2. STANDARD TRACEROUTE
The Time To Live (TTL) field in the IP header plays a central role for traceroute. Each IP packet has a TTL that is decremented by every router before forwarding it. In case the TTL reaches the value of zero, the router will drop the packet instead of forwarding it and send an error message (Time Exceeded, TTL Expired) using the Internet Control Messages Protocol (ICMP) back to the original sender. Dropping packets after they have stayed in the network too long is crucial – otherwise corrupt routing tables could cause a packet to be endlessly routed in a loop, thereby congesting the network.

2.1 Discovering paths
In order to discover the path to a destination D, traceroute exploits the TTL field with the following algorithm:

1. $n := n + 1$
2. Send a packet with TTL n to D
3. Wait for an ICMP error message
4. Save the error message's source IP

The above procedure starts with $n = 0$ and is repeated until a reply from D is received.

The first packet sent by the initiator will have TTL 1. That means the TTL will expire on the first router on the path to D. Assuming this first router isn't already the desired destination D, it will drop the message as described above and return an ICMP error (Time Exceeded, TTL Expired) to the initiator. Now, knowing the first router on the way to D, the initiator sends the next packet (according to the algorithm) with TTL 2. This will yield the IP address of the second router on the way to D.

The whole process goes on, until the received ICMP message actually comes from D. This will usually also be an ICMP error message (Port Unreachable), because most likely D won't be listening on the destination port we were using for that packet.

2.2 Small variations
The above algorithm to discover paths isn't completely specified yet. Most notably, the following details were left out:

- The transport layer protocol used for the package sent to D in the first step
- The destination port used in the first step

Typical traceroute implementations use the User Datagram Protocol (UDP) together with a non-reserved, arbitrary destination port. This port is increased for each subsequent packet in order to relate the sent packets to their responses.

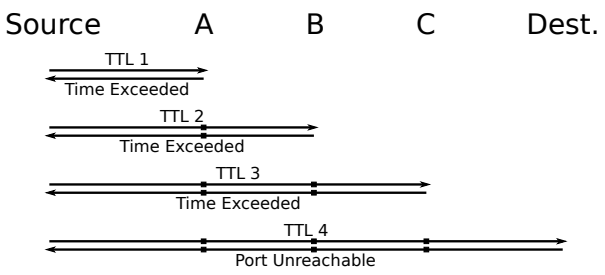


Figure 2: A traceroute example

Technically speaking, it does not matter if the Transmission Control Protocol (TCP) or ICMP is used instead of UDP for sending the packets. The port number may also be chosen from the pool of the well-known ports and doesn't need to be varied for each different packet. However, the packets have to be different somewhere within the first 8 octets of the transport protocol's header, otherwise establishing a relation between the sent packet and the ICMP response (holding only the first 8 octets of the transport protocol's header) can't be established.

Varying transport protocols and ports may seem insignificant, but they play a very important role, for example, when trying to trace through a firewall (discussed in section 3.2).

2.3 An example

Figure 2 shows a trace of four hops to the destination. The traceroute output for that example would look like this:

Traceroute to Dest (Dest.IP)

1	A (A.IP)	x_A ms	y_A ms	z_A ms
2	B (B.IP)	x_B ms	y_B ms	z_B ms
3	C (C.IP)	x_C ms	y_C ms	z_C ms
4	Dest (Dest.IP)	x_{Dest} ms	y_{Dest} ms	z_{Dest} ms

In this example output, the letters A, B and C denote the DNS names of the 3 routers that were discovered. With .IP appended, they denote the respective IP address. For each of the routers, three times in milliseconds (ms) are given. To reduce the chance of failure, traceroute actually sends three packets with the same TTL (thus reaching the same router) and measures their round trip time. These three times are given as x, y and z.

3. PROBLEMS

Since traceroute wasn't really the intended use of the TTL field, this approach of finding paths introduces some problems.

3.1 Non-standard routers

One major problem for traceroute is caused by routers that disregard RFC 1812 [3]. In this paper, three common problems caused by such non-standard routers will be discussed.

3.1.1 Unknown routers

This problem exists due to routers discarding packets, without sending an ICMP Time Exceeded message back to the origin. When traceroute tries to discover such a router, it won't receive a response. The result in the output of traceroute will be a star (*) instead of the DNS name and IP address of the router in question.

3.1.2 Missing routers

Routers not decrementing the TTL field at all cause this problem. When the TTL reaches the appropriate value to discover such a router, the next router in the path will instead be discovered. This will result in a router completely missing in the trace. Even worse: There is no way of knowing that the packet has passed an additional router.

3.1.3 Loops in trace

We will call the appearance of the same router twice in the same trace a loop. Routers causing this problem decrement the TTL as they should but fail to discard packets that have reached a TTL of zero. When the TTL of the probe packet reaches the right value to discover this router, the packet will be forwarded once more and the next router on the path (N) will be discovered instead. When traceroute increases the TTL by one, it will again discover the router N, which correctly drops the packet once again. In the resulting trace, the router N appears twice in a row while the non-standard router causing this problem is missing.

3.2 Firewalls

Firewalls play an important role in today's Internet. Almost every corporate network will operate at least one firewall as gatekeeper to their network. As operating systems and applications are – due to their complexity – error prone, using firewalls plays a vital role in security. Even though traceroute itself should not be regarded as an attack, it may give a potential attacker valuable insight into the network internals. Since this information can be used for planning and carrying out attacks, some network administrators deliberately block traceroute with their firewall policy.

Two types of problems when dealing with firewalls will be discussed.

3.2.1 Blocked protocols and ports

A very common way of protecting a network is to limit incoming traffic to certain combinations of protocol and port depending on the destination. A company operating a web server within their corporate network, for example, needs to allow incoming traffic to that particular host on TCP port 80.

This problem can be circumvented by trying different protocol and port combination, as described in Section 2.2. Eventually a combination allowed by the firewall policy will be found and a path to the destination within the firewall protected network can be detected.

3.2.2 Blocked ICMP time exceeded

Firewalls blocking ICMP packets of the type time exceeded deliberately block attempts to gain information about the network with traceroute. Since the ICMP time exceeded packets generated from within the network can't reach the initiator of the trace, there is no way to gain information about the network beyond such a firewall.

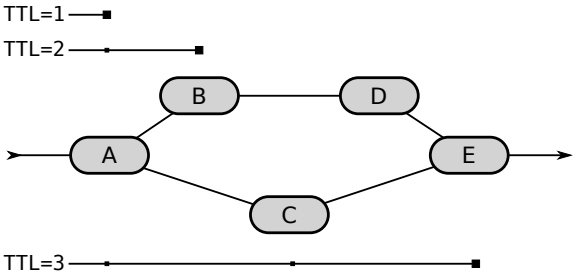


Figure 3: Problem caused by load balancers

3.3 Load balancers

Networks are usually designed to deal with failures of single points without loosing connectivity. This implies, that in most cases multiple possible paths between two distinct nodes exist. In the Internet, routers automatically update their routing tables when they realize that a certain connection is no longer available.

In order to increase performance of the network, backup routes are not only used to avoid fatal failures but also to provide extra bandwidth by distributing the packets over all available routes to the desired destination. [2, 11]. This behavior makes traceroute's life hard: As one trace consists of multiple packets that are individually balanced over the available paths, traceroute may actually detect non-existent links.

An example is given in Figure 3: Node A is a load balancer and in order to reach our desired destination we have to pass router E. For simplicity reasons we assume that A is already our next hop router, reachable with TTL 1. For the second packet (sent with TTL 2), the load balancer decided to forward the packet to router B. The third packet (TTL 3) was sent via router C. The resulting route to E looks like this:
A $\rightarrow$ B $\rightarrow$ E
We will wrongly assume that there exists a link between the routers B and E.

Dealing with this problem is difficult. Traceroute uses 3 packets for each single router to discover. This ensures that it will likely be noticed that something unexpected is happening since packets sent with the same TTL result in responses from different IP addresses. Unfortunately there is no way to decide on a correct path based on the available information. Nevertheless, some traceroute-like implementations that will be discussed in Section 4 have come up with ideas to deal with this problem.

4. DIFFERENT APPROACHES

In this section different route discovery approaches will be introduced. Most important, the differences from standard traceroute will be pointed out and discussed.

4.1 Nmap traceroute

The well known port scanner nmap (Network MAPper) brings it's own traceroute implementation. That way, portscanning and tracing target hosts can be conveniently combined. While nmap uses basically the same approach as standard traceroute, it starts the trace with a high TTL decreasing it step by step to 1 instead of starting with 1 and increasing. This feature is meaningful when tracing whole subnetworks: Once the performed backward trace reaches a point that was passed in a previous trace, there is no need to complete the whole trace any more since the path from that point to the initiator is already known. Thus, for tracing whole networks, the amount of packets required is significantly reduced. For single hosts, this attempt is a little bit slower, because the correct highest TTL value has to be discovered.

4.2 Traceroute-nanog

Traceroute-nanog is a route discovery application from the North American Network Operators' Group (NANOG). This is just another standard traceroute implementation with the following additions:

- Option to lookup Autonomous System (AS) numbers for each hop.
- Change the Type Of Service (TOS) field in the IP header of sent packages to arbitrary values.
- Option to perform a Maximum Transmission Unit (MTU) discovery along the path being traced.
- Can detect the use of the Multiprotocol Label Switching (MPLS) protocol (application described in [4])
- Offers a parallel mode: In order to speed up the trace, multiple packets are sent at the same time. Since some routers limit the amount of ICMP packets per minute, setting this value too high may result in packet loss.

4.3 Traceroute using an IP Option

In an attempt to simplify traceroute, the Internet Engineering Task Force published RFC 1393 [6], introducing an IP option for traceroute. Using this IP option, traceroute is performed in the following way:

1. The initiator sends a packet to the desired destination with the IP traceroute option set.
2. Every router receiving the packet will not only forward it but additionally send a newly generated extra packet back to the initiator.

This approach has two advantages. Firstly, the amount of packets needed for the whole trace is $n+1$ packets (with n denoting the length of the path). The total number of hops taken by all the packets is $n + \sum_{i=1}^{n} i = \frac{(n+1)^2+n-1}{2}$. Using traceroute, the number of packets required is $2n$ (without doing multiple measurements per router) and the number of total hops is $2 \cdot \sum_{i=1}^{n} i = (n+1)^2 - n + 1$. This comparison is actually in favor of traceroute, because the main reason for doing multiple measurements is not averaging the round trip times. It's all about at least detecting problems described in Chapter 3.

Secondly, the one packet traveling from the source to the destination while triggering an extra packet from each router

travels through the network on just one (naturally consistent) path. While there's no way to notice load balancers on the way, the resulting path is always valid and all detected links actually exist.

Unfortunately, this option isn't implemented on any public router. This is mainly due to the security risk such an option represents: A malicious user could trigger a large number of packets sent to a particular host by sending just one packet with forged source address to a far away destination. It is very unlikely that we will see such a convenient way to discover paths in the future.

4.4 Paris traceroute

Paris traceroute's [1] main goal is to do better in the presence of load balancers. The programmers use the fact, that most load balancers use a per-flow approach in their load balancing decision: packets from the same flow are forwarded along the same path. This implementation's main achievement is to control the packet header fields of the probe packets, thereby enabling them to be detected as part of the same flow and routed along the same path by per-flow load balancers.

Through experimentation, the scientists discovered that, apart from some fields in the IP header – namely the TOS, protocol, source address and destination address – the first four octets of the transport layer header are used to identify a packet as being part of a flow. On the other hand, only the first eight octets of that header are encapsulated in the ICMP time exceeded message. So, in order to establish a relation between the probe packets and the ICMP errors, we need to perform modifications in octets 5 to 8 of the transport layer header.

This is easiest when using the TCP protocol: the first 4 octets of the TCP header consist of the source and destination port, the second 4 octets hold the 32 bit sequence number. Thus, varying only the sequence number while keeping source and destination port constant allows all the packets to be regarded as the same flow.

Using UDP things get slightly more difficult: The first four octets also hold source and destination port, but the second four octets hold the length and checksum of the packet. Since neither length nor the checksum can be changed independently from the payload (otherwise the packet is liable to be discarded because of an incorrect checksum), Paris traceroute actually varies the payload in order to produce different checksums. The value of those checksums is ultimately the information used to relate the ICMP time exceeded messages with the sent probe packets.

Using ICMP for sending the probes is the most challenging: The header's first four octets hold the type, code and checksum header fields. The second four octets consist of the identifier and sequence number fields. In order to allow the packets to be identified as part of the same flow, Paris traceroute varies both the Identifier and Sequence number fields in such a way, that the computed checksum remains constant.

According to the authors, Paris traceroute does significantly better than traditional traceroute. According to their paper [1], the number of loops observed was reduced by approximately 84%. Also, this variant of traceroute is less likely to report non-existent links as explained in Section 3.3. The number of faulty links reported is reduced by about 64%.

4.5 DisCarte

DisCarte [8] – standing for Disjunctive Internet Cartographer – is the most advanced route discovery program covered in this paper. It is really more than just another traceroute utility: It can be used to generate topology maps of networks. DisCarte makes good use of traditional traceroute together with an additional feature of the IP protocol: the Record Route (RR) option.

4.5.1 The record route option

The record route option as specified in RFC 791 [7] is older than the IP traceroute option discussed in Section 4.3. It works as follows:

1. The initiator sends a packet with the RR option set to the desired destination.
2. Every router on the path will, while forwarding the packet, add it's own IP address to the IP header.

This approach has a major drawback: The amount of space in the IP header is limited – only up to 9 addresses may be recorded with the record route option. After nine addresses have been added, subsequent routers will just forward the packet without performing any manipulation. This is not as bad as it may seem at first, since using a geographically distributed set of starting points usually offers one point within (or at least close to) the distance of nine hops from the destination.

Surprisingly, only very few routers – according to [8], just a little bit more than 1% – actually filter packets with the RR option set. At the same time, due to the fact that RR is under-standardized, router implementations vary in the way they treat packets with RR set:

- NotImpl: About 9% of the routers don't implement this option at all.
- Departing: About 62% update the RR array at the time packets leave the router, adding the IP address of the router's outgoing interface. This implies that packets arriving with TTL 1 will be dropped before the RR array is updated.
- Arriving: A rather small number of the routers, approximately 7%, already update the RR array at the time the packet arrives, putting the IP address of either the router's outgoing interface or the router's internal loopback interface into the RR array. Those routers also update the RR array of packets that are about to expire.

[8] introduces even more classes of different behavior in respect of the RR option, but introducing them all would go beyond the scope of this paper.

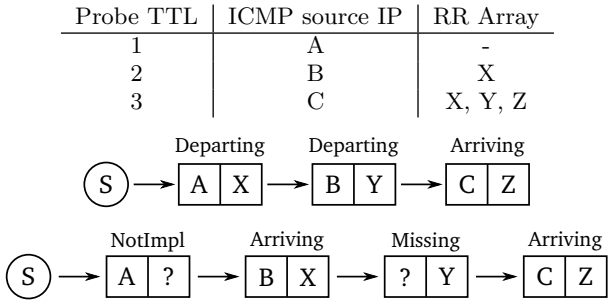


Figure 4: Example taken from [8] showing how the combination of different RR implementations and wrong behavior with respect to the TTL parameter may yield ambiguous results.

4.5.2 Combining RR and traceroute

The idea of combining record route with traceroute seems to be simple enough. In practice, combining the information in a meaningful way becomes quite challenging: ICMP time exceeded messages usually contain the address of the router's incoming interface while the RR array mostly contains the address of the router's outgoing interface (see Section 4.5.1). Moreover, due to some abnormalities caused by non-standard routers as described in Section 3.1, the routers discovered by RR and traceroute may differ. For that reason, the actual architecture of the path often becomes ambiguous.

The example in Figure 4 illustrates such a case: The table on top shows the information gained by both the IP address of the returning ICMP packet (traditional traceroute) and the content of the RR array. A, B, C denote the router's incoming and X, Y, Z the router's outgoing interface respectively. From the information given in the table, at least two different topologies for the path can be inferred, illustrated by the two paths drawn. Arriving, Departing and NotImpl refer to the different possible record route implementations as described in Section 4.5.1. Missing refers to the non-standard router implementation introduced in Section 3.1.2.

In order to cross-reference the results, DisCarte uses Disjunctive Logic Programming: Using the results from both the traceroute and the RR trace, all possible paths that do not contain the same router twice in a row are generated. The most likely path (according to the likeliness of each router type) is then selected. In Figure 4 it is easy to see that the upper (smaller) path would be selected, because the combination (Departing, Departing, Arriving) has a higher probability than (NotImpl, Arriving, Departing).

It should be noted, that only a small part of DisCarte was described – the discovery of one single route. The authors of DisCarte evaluate their program against other cartographers (namely Passenger [9] and Rocketfuel [10]) and not against any of the pure traceroute utilities explained in this paper. Therefore no direct comparison and evaluation can be offered at that point. However, according to [8], DisCarte performs significantly better than Rocketfuel, which relies on pure traceroute for route discovery.

5. CONCLUSION

Many scientists have tried dealing with the problem of finding routes and many different implementations for route discovery are available. Nevertheless, taking a closer look, most of these implementations are not very different from standard traceroute. And, just like standard traceroute, all of these implementations have problems when dealing with non-standard situations. Still – over 20 years after the first traceroute application was implemented by Van Jacobson [5], traceroute remains a difficult problem.

6. REFERENCES

[1] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira. Avoiding traceroute anomalies with paris traceroute. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 153–158, New York, NY, USA, 2006. ACM.

[2] B. Augustin, T. Friedman, and R. Teixeira. Measuring load-balanced paths in the internet. In *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 149–160, New York, NY, USA, 2007. ACM.

[3] F. Baker. Requirements for IP version 4 routers. RFC 1812, Internet Engineering Task Force, 1995. Available from: `http://www.ietf.org/rfc/rfc1812.txt`.

[4] R. Bonica, D. Gan, D. Tappan, and C. Pignataro. ICMP Extensions for Multiprotocol Label Switching. RFC 4950, Internet Engineering Task Force, 2007. Available from: `http://www.ietf.org/rfc/rfc4950.txt`.

[5] V. Jacobson. Traceroute [online]. 1989. Available from: `ftp://ftp.ee.lbl.gov/traceroute.tar.gz`.

[6] G. Malkin. Traceroute using an IP option. RFC 1393, Internet Engineering Task Force, 1993. Available from: `http://www.ietf.org/rfc/rfc1393.txt`.

[7] J. Postel. Internet Protocol. RFC 791, Internet Engineering Task Force, 1981. Available from: `http://www.ietf.org/rfc/rfc791.txt`.

[8] R. Sherwood, A. Bender, and N. Spring. Discarte: a disjunctive internet cartographer. In *SIGCOMM '08: Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, pages 303–314, New York, NY, USA, 2008. ACM.

[9] R. Sherwood and N. Spring. Touring the internet in a TCP sidecar. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 339–344, New York, NY, USA, 2006. ACM.

[10] N. Spring, R. Mahajan, and D. Wetherall. Measuring isp topologies with rocketfuel. *SIGCOMM Comput. Commun. Rev.*, pages 133–145, 2002.

[11] D. Veitch, B. Augustin, R. Teixeira, and T. Friedman. Failure control in multipath route tracing. In *INFOCOM*, pages 1395–1403. IEEE, 2009.

Mercator

Florian Hartmann
Betreuer: Johann Schlamp, Dirk Haage
Hauptseminar - Innovative Internettechnologien und Mobilkommunikation SS2010
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: hartmanf@in.tum.de

KURZFASSUNG

Netzwerke, insbesondere das Internet, unterstehen einer ständigen Veränderung. Die Übersicht über ein Netzwerk zu behalten, ist schwierig. Programme sind in der Lage, Karten eines Netzwerkes durch Messungen zu erstellen. Diese können in diversen Anwendungsgebieten hilfreich sein. In dieser Arbeit wird das Programm Mercator beschrieben, das durch Traceroute-Messungen im Internet Daten liefert, mit denen später eine Internet-Karte erstellt werden kann. Es werden die genutzten Techniken erklärt, die zum Auffinden von Routern und Verbindungen, sowie zum Zuordnen von Interfaces zu Routern nötig sind. Darüber hinaus wird auf Ergebnisse von Testmessungen eingegangen und mögliche Verbesserungen der Techniken aufgezeigt.

Schlüsselworte

Internet-Karten, Netzwerktopologie, Mercator, Traceroute, Source-Routing, Alias-Auflösung, hop-limited probes, informed random address probing

1. EINLEITUNG

Das Internet besteht aus vielen untereinander verbundenen Computernetzwerken. Zu diesen Netzwerken gehören hauptsächlich Provider-, Firmen- oder Universitäts- bzw. Forschungsnetzwerke. Jedes dieser Netzwerke steht unter einer eigenen administrativen Verwaltung. Da die Infrastrukturen der einzelnen Netzwerke unabhängig voneinander sind, kann jede Verwaltung eigene Regeln und Richtlinien festlegen. Dies hat zur Folge, dass Netzwerke unterschiedlich aufgebaut sein können und daher oftmals unterschiedliche Netzwerktopologien besitzen. Des Weiteren unterstehen Netzwerke einem stetigem Wandel. Teilnehmer und Verbindungen kommen hinzu, werden entfernt oder Eigenschaften ändern sich.

Den Überblick über ein Netzwerk zu behalten, ist eine schwierige Aufgabe. Zum einen ist es eine zeitaufwändige Angelegenheit, eine Netzwerktopologie per Hand zu pflegen, und zum anderen geben die Verwaltungen oftmals keine detaillierten Informationen über ihr Netzwerk preis. Dennoch können genaue Informationen über Netzwerktopologien, wie in Kapitel 3.2 beschrieben, in vielerlei Hinsicht hilfreich sein. In Kapitel 3 wird erläutert, was Internet-Karten sind und wie diese aussehen können.

Das Programm Mercator ist in der Lage, die Topologie eines Netzwerkes herauszufinden, ohne dabei Informationen über das entsprechende Netzwerk zu benötigen. Durch die dabei entstehenden Daten lassen sich Internet-Karten generieren. In Kapitel 4 wird dieses Programm, sowie dessen Techniken vorgestellt. Weiterhin wird auf die bei Messungen entstandenen Ergebnisse eingegangen und diese bewertet. In Kapitel 5 wird ein Ausblick auf weitere Techniken gegeben, die ebenfalls zur Generierung einer Internet-Karte genutzt werden können.

2. VERWANDTE TOOLS

Neben Mercator gibt es noch weitere Tools, die Internet-Karten erstellen können.

Rocketfuel [13], ein Tool der Universität von Washington (USA), ist darauf ausgelegt, die Netzwerktopologie eines gegebenen Internet Service Providers abzubilden. Rocketfuel läuft auf knapp 300 Servern in Europa, Australien und der USA.

Ein weiteres Tool ist Skitter [7]. Skitter wird von der Cooperative Association for Internet Data Analysis, kurz CAIDA, betrieben und hat innerhalb der letzten zehn Jahre über vier Terabyte Daten gesammelt, die Grundlage vieler Visualisierungen sind.

Auch scamper [11] ist ein Tool zur Erstellung von Internet-Karten. Es wird von der WAND group der Universität von Waikato (Neuseeland) entwickelt. Scamper nutzt aktuellere Techniken als Mercator und unterstützt unter anderem auch IPv6. Es wird von CAIDA in einer neuen Mess-Infrastruktur genutzt, die Skitter ablösen soll.

3. INTERNET-KARTEN

Von einem Netzwerk können Karten generiert werden. Dabei wird in der Regel zwischen drei verschiedenen Arten von Karten unterschieden [8].

- Geografische Karten zeigen die geografische Position der Teilnehmer auf einer Landkarte. Solche Karten können genutzt werden um Netzwerkprobleme geografischer oder klimatischer Natur zu lösen.
- Konzeptionelle Karten hingegen betrachten nicht den physikalischen Aufbau, sondern die Verteilung der Informationen im Netzwerk. Diese Karten können genutzt werden, um Informationen im Netzwerk zu finden. Generiert und genutzt werden solche Karten vor allem von Suchmaschinen wie Google oder Yahoo. Durch Verbindungen zwischen verschiedenen Informationen lässt sich die Relevanz der Daten bemessen.

- Eine weitere Art von Karten sind die Infrastrukturkarten. Bei diesen Karten handelt es sich um die Darstellung einzelner Netzwerkteilnehmer, Knotenpunkte und deren Verbindungen untereinander. Entsprechende Anwendungsgebiete solcher Karten werden im Kapitel 3.2 betrachtet.

3.1 Definition

Im Folgenden wird mit Internet-Karte immer eine Infrastrukturkarte, wie im Kapitel 3 beschrieben, bezeichnet.

Eine Internet-Karte besteht aus einem ungerichtetem Graphen. Die Knoten dieses Graphens entsprechen den Routern bzw. Hosts im Internet. Dabei kann ein Router mehrere Interfaces besitzen. Jedes Interface ist einer IP-Adresse zugeordnet, wobei diese Adresse im Netzwerk eindeutig ist. Die Knoten sind über Kanten miteinander verbunden. Eine Kante entspricht einer logischen Verbindung zweier Interfaces. Ein Interface ist immer mit einem oder mehreren Interfaces verbunden. Die Verbindung ist auf IP Ebene zu sehen - Repeater, Bridges, Hubs oder Switches werden vernachlässigt. Abbildung 1 zeigt eine mögliche graphische Darstellung einer Internet-Karte.

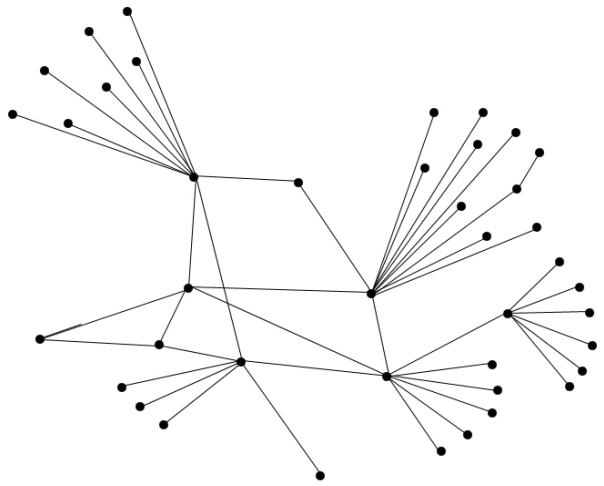


Abbildung 1: Darstellung einer Internet-Karte

3.2 Anwendungsgebiete

Internet-Karten können in vielen Szenarien hilfreich sein. Zur Simulation von Netzwerken können Internet-Karten die benötigten Daten liefern. Der Vorteil besteht darin, dass die Informationen auf realen Netzwerken basieren und somit realistische und praxisnahe Simulationen zulassen [5]. Beispielsweise werden in Referenz [12] simulierte Netzwerke genutzt, um Annahmen über die Skalierung von Multicast-Netzwerken zu bestätigen.

Karten eines Netzwerkes sind ebenfalls für dessen Verwaltung nützlich. Die Netzwerktopologie kann bei der Platzierung neuer Knotenpunkte helfen. Zudem können Probleme und Flaschenhälse identifiziert und entsprechend aufgelöst werden [1].

Nicht nur für die Netzwerkbetreiber bzw. die Verwaltung ist es interessant, ihre Netzwerktopologie zu kennen. Auch Teilnehmer des Netzwerkes können anhand der Karte entscheiden, wo sie sich im Netzwerk platzieren wollen und welcher Internetprovider dafür gewählt wird. Besonders im Internet ist es wichtig, Server richtig aufzustellen und somit die Latenzzeit zur Zielgruppe zu minimieren und die verfügbare Bandbreite zu maximieren [1].

Ist in einem Netzwerk die komplette Topologie bekannt, können spezielle Protokolle und Algorithmen angewandt werden. Es existieren Routingprotokolle, die anhand der Netzwerktopologie den kürzesten Weg zum Ziel berechnen können (beispielsweise das Optimized Link State Routing Protokoll, kurz OLSR). Allerdings sind solche Routingprotokolle nur für kleine bzw. statische Netzwerke nützlich. Für große, stark fluktuierende Netze wie das Internet, sind solche Algorithmen kaum einsetzbar.

Es gibt demnach vielfache Anwendungsmöglichkeiten für Internet-Karten. Im folgenden Kapitel wird das Programm Mercator vorgestellt, das die für Internet-Karten benötigten Informationen durch Messungen herausfinden kann.

4. MERCATOR

Mercator ist ein Programm, das 1998 von Ramesh Govindan und Annop Reddy am Information Sciences Institute an der University of Southern California (USA) entwickelt wurde. Das Programm ist in der Lage, Informationen durch Messungen im Internet zu liefern, die für die Erstellung von Internet-Karten vonnöten sind. Genau genommen sucht Mercator Router im Netzwerk und Verbindungen zwischen den Routern. Des Weiteren kann Mercator Aliasse auflösen, also Interfaces zu Routern zuordnen.

4.1 Anforderungen

Da Mercator Informationen über die Infrastruktur des Internets liefern soll, gibt es bestimmte Einschränkungen, die das Programm beachten muss. Im Allgemeinen bietet das Internet nur wenige Funktionen zur Infrastrukturbestimmung. Protokolle zur Netzwerkverwaltung, wie zum Beispiel das Simple Network Management Protocol, sind in der Regel nicht verfügbar. Für Mercator wurden demnach die drei folgende Anforderungen gewählt [9].

- Die Anforderungen an das Netzwerk sollen minimal sein. Das bedeutet, Mercator nutzt zur Informationsgewinnung nur Techniken, die standardmäßig jeder Router im Internet unterstützt. Im Grunde basieren Mercators Messungen nur auf Paketen, deren time-to-live begrenzt ist, wie auch beim Programm Traceroute.

- Die Messungen sollen von einem einzelnen, frei wählbaren Knoten im Netzwerk ausgeführt werden können. Durch eine einzelne Instanz des Programmes auf einem Knoten wird die Programmierung und der Betrieb vereinfacht. Ein komplexes, verteiltes System ist nicht nötig. Mercator soll außerdem auf einem beliebigen Host im Netzwerk installiert werden können. Denn eine Platzierung auf Transit- oder Hauptknotenpunkten ist oftmals nicht möglich oder durch die Netzwerkverwaltung nicht gewünscht.

- Router und deren Verbindungen werden von Mercator durch Messungen bestimmt. Diese Messungen werden zu bestimmten Ziel-Adressen ausgeführt. Die Ziele könnten durch Adresstabellen definiert werden. Adresstabellen müssten dann allerdings aufwändig per Hand gepflegt werden und könnten dementsprechend fehlerhaft und nicht immer aktuell sein. Die Wahl der Ziele muss deshalb vom Programm selbst getroffen werden.

4.2 Ziele

Neben dem Hauptziel, Daten für eine Internet-Karten zu beziehen, wurden noch weitere Ziele für Mercator definiert. So soll das Programm seine Messungen möglichst effektiv durchführen. Das bedeutet, dass die Anzahl der Pakete, die gleichzeitig und insgesamt gesendet werden, möglichst klein sein soll. Somit soll eine geringe Netzwerkbelastung erreicht werden [9].

Weiterhin sollen die Messungen schnell durchgeführt werden. Dies steht aber in Konkurrenz zur geringen Netzwerkbelastung. Denn schnelle Messungen erfordern eine große Menge an parallel versendeten Paketen. Ziel ist es also, die richtige Balance zwischen Effektivität und Geschwindigkeit zu erzielen.

Die gefundenen Informationen über das Netzwerk sollen darüber hinaus vollständig und fehlerfrei sein. Wie in Kapitel 4.3.4 beschrieben, können aber möglicherweise nicht alle Router und Verbindungen gefunden werden, wenn die Messungen nur von einem Knotenpunkt im Netzwerk ausgeführt werden.

4.3 Techniken

Um die festgelegten Ziele zu erreichen, setzt Mercator verschiedene Techniken ein. Dabei werden Messungen, so genannte „Traceroutes“, zu bestimmten IP-Adressen durchgeführt. Die Messungen entdecken und speichern Router und Verbindungen zwischen den Routern. Genau genommen werden allerdings keine Router gefunden, sondern nur einzelne Interfaces mit den dazugehörigen IP-Adressen. Zusätzlich zu den genannten Messungen führt Mercator weitere Tests durch, die die gefundenen Interfaces zu ihren Routern zuordnen sollen. Letztendlich ergeben sich Router, ihre zugehörigen Interfaces sowie die Verbindungen zwischen den Routern. Mit diesem Ergebnis ließe sich dann eine Internet-Karte generieren. Dies ist aber nicht Teil von Mercator.

4.3.1 Traceroute

Die Mercator zu Grunde liegende Technik nennt sich „Traceroute“. Bei einem Traceroute wird der Pfad vom Messknoten zu einem Zielknoten herausgefunden. Das heißt, alle Knoten zwischen Start und Ziel werden nacheinander durchlaufen. Dabei werden sogenannte „hop-limited probes“ genutzt, dies sind Pakete deren time-to-live, folgend als TTL abgekürzt, begrenzt ist. Die TTL bezeichnet die Anzahl an Knotenpunkten, folgend als Hop bezeichnet, die ein Paket traversieren darf, bevor es verworfen wird. Jeder Hop dekrementiert die TTL um eins. Sobald der Wert auf Null sinkt, sendet der aktuelle Hop eine ICMP-time-exceeded-Fehlermeldung an den Sender zurück. Sollte der aktuelle Hop das Ziel des Paketes sein, wird eine ICMP-echo-reply-Meldung zurück gesendet. ICMP-Meldungen (Internet Control Message Protocol) sind Bestandteil von IPv4 und werden zum Austausch von Informations- und Fehlermeldungen genutzt. Die hier beschriebene Technik wird auch vom Programm Traceroute genutzt, das auf gängigen System verfügbar ist (Unix/Mac: traceroute, Windows: tracert.exe).

Bei „traceroute“-Messungen können TCP-, UDP- oder ICMP-Pakete verwendet werden. Als erstes wird ein Paket von Mercator zum Zielknoten versendet dessen TTL = 1 gesetzt ist. Der erste Hop (A) auf der Route zum Ziel verringert die TTL, somit ist die TTL = 0, der Hop verwirft das Paket und sendet eine ICMP-time-exceeded-Fehlermeldung an den Startknoten zurück. Anhand der Quelladresse der ICMP-Fehlermeldung kennt Mercator nun die Adresse von A und weiß außerdem, dass A direkt mit ihm verbunden ist.
Jetzt wird ein zweites Paket mit der TTL = 2 versendet. Hop A verringert die TTL auf 1 und sendet das Paket weiter. Diesmal dekrementiert Hop B die TTL auf Null. B antwortet nun ebenfalls mit einer ICMP-time-exceeded-Fehlermeldung. Mercator kann dann wiederum anhand der Quelladresse des ICMP-Paketes B identifizieren und weiß, dass A mit B direkt verbunden ist.
Mercator versendet nun so lange neue Pakete, jedes mal mit höherer TTL, bis eine ICMP-echo-reply-Meldung zurück kommt. In diesem Fall weiß Mercator, dass der Zielpunkt erreicht wurde und die Messung abgeschlossen ist. Das Ergebnis ist der komplette Pfad von Mercator bis zum Endpunkt inklusive aller Hops und deren IP-Adressen. Mercator verschickt immer nur ein Paket pro Traceroute-Messung gleichzeitig. Parallele Messungen sind jedoch möglich.

Mit dieser Technik kann Mercator Knotenpunkte und deren Verbindungen untereinander in einem Netzwerk herausfinden und schafft somit die Grundlage für Internet-Karten.

4.3.2 Adresswahl

Im letzten Kapitel wurde erläutert, dass Mercator seine Traceroute-Messungen zu bestimmten Zielknoten durchführt. In Kapitel 4.1 wurde an Mercator die Anforderung gestellt, die Adressen der Zielknoten selbstständig zu generieren. Des Weiteren ist gefordert, dass die entstehende Karte vollständig ist. Der einfachste Weg wäre eine Messung zu allen mögliche IP-Adressen zu starten. Allerdings umfasst der IPv4-Adressbereich über vier Milliarden mögliche Ziele. Eine Messung zu jeder IP-Adresse auszuführen würde viel zu lange dauern. Außerdem sind viele dieser IP-Adressen nicht erreichbar und eine Messung zu diesen Adressen würde wenig neue Informationen bringen. Folglich kann nur eine begrenzte Anzahl an Messungen durchgeführt werden. Ziel ist es deshalb, nur Teilnetze zu wählen und diese dafür vollständig abzudecken.

Mercator nutzt dafür eine Technik namens „informed random address probing“. Dabei wird anhand von zwei Annahmen der Adressbereich eingeschränkt. Der Bereich wird so gewählt, dass er große Mengen an adressierbaren, also erreichbaren IP-Adressen enthält. Ein Adressbereich bzw. Block wird durch ein Präfix dargestellt (z.B.: 128.10/16). Alle IP-Adressen aus einem Block haben ein gemeinsames Präfix. Mercator wählt 8, 16 oder 19 Bit als Länge des Präfixes. Diese Längen wurden so gewählt da Netzwerkbetreiber oftmals diese Längen für ihre Präfixe verwenden [9].

Die erste Annahme geht davon aus, dass IP-Registrierungs-

stellen ihre IP-Adressen immer in Blöcken verteilen [9]. Der Besitzer eines solchen Adressblockes wird also zuerst einen Großteil der IP-Adressen in diesem Block vergeben, bevor er einen neuen Adressblock anfordert. Mercator geht daher davon aus, dass wenn in einem Adressblock eine erreichbare IP-Adresse vorhanden ist, noch weitere adressierbare IP-Adressen in diesem Block existieren. Beim Programmstart wählt Mercator den Adressblock, in dem es sich selbst befindet, als Initialblock. Dann werden so lange zufällige IP-Adressen aus diesem Block gewählt bis über eine Zeitspanne von drei Minuten keine neuen erreichbaren IP-Adressen gefunden werden können.

Wenn in einem Adressblock keine neuen IP-Adressen mehr gefunden werden, muss Mercator einen neuen Block wählen. Dazu wird die zweite Annahme verwendet. Diese besagt, dass IP-Registrierungsstellen ihre Adressblöcke sequentiell vergeben. Existieren also in einem Block adressierbare IP-Adressen, so werden höchstwahrscheinlich auch in den Nachbarblöcken erreichbare IP-Adressen existieren [9]. Beispielsweise sind Block 128.9/16 und 128.11/16 Nachbarblöcke von 128.10/16. Um einen neuen Block aus allen Nachbarblöcken zu wählen, wird ein Lotterie-Scheduling-Verfahren angewandt. Dabei ist die Anzahl an Lottoscheinen proportional zu der gefundenen Anzahl an IP-Adressen im jeweiligen Nachbarblock.

Beide Annahmen helfen Mercator, weniger Messungen zu nicht erreichbaren IP-Adressen durchzuführen und ermöglichen letztendlich eine schnellere Erforschung des Netzwerkes. Außerdem liegen die gefundenen Adressen sehr nahe beieinander, was zu einem relativ kompletten Teilnetz führt.

4.3.3 Reduzierung benötigter Pakete

Auf Grund der Wahl der Ziele (Kapitel 4.3.2) werden sehr viele Traceroute-Messungen pro Block ausgeführt. Wie in Abbildung 2 zu sehen, ist der Pfad zum Block 128.10/16 immer der selbe. Bei jeder Messung den gesamten Pfad zu traversieren, würde keine neuen Informationen liefern. Daher reicht es, wenn Mercator die TTL des ersten Paketes so wählt, dass dieses den ersten Knoten im Block erreicht. Kommt die resultierende ICMP-time-exceeded-Fehlermeldung von dem erwarteten Knoten, führt Mercator seine Messung von dort an fort [9].

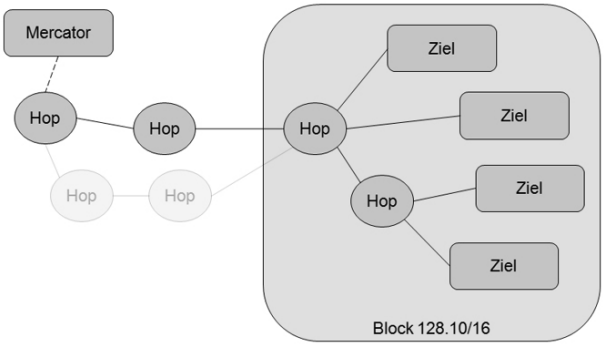


Abbildung 2: „traceroute“-Messung zu einem Adressblock

Unter Umständen kann sich der Pfad zum ersten Knoten im Block verändern, siehe Abbildung 3. Dies geschieht zum Beispiel durch alternative Pfade oder durch Ausfall von Verbindungen und Routern. In diesem Falle merkt Mercator aber anhand der Quelladresse der ersten ICMP-Fehlermeldung, ob der Knoten dem erwarteten entspricht. Sollte dies nicht der Fall sein, prüft Mercator den Pfad rückwärts bis es zum ersten Knoten stößt, der ihm bekannt ist. Von diesem Knoten an kann es dann seine Traceroute-Messung ganz normal weiterführen. Durch diese Technik wird die Gesamtanzahl benötigter Pakete verringert.

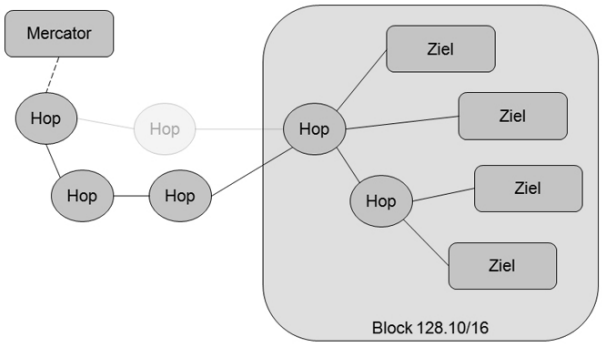


Abbildung 3: Änderung des Pfades zum Block

4.3.4 Source-Routing

Mit den bisher vorgestellten Techniken lassen sich viele Knotenpunkte und Verbindungen zwischen Knoten herausfinden. Doch können trotzdem Pfade unbekannt bleiben. Wie in Abbildung 4 zu sehen kann es passieren, dass durch Messungen zwar die Hops A und B gefunden werden, Pakete aber nie die Verbindung zwischen A und B passieren. Dies liegt an der Entscheidung, Mercator nur auf einem einzigen Knoten im Netzwerk laufen zu lassen (siehe Kapitel 4.1). Könnte eine weitere Instanz von Mercator auf Hop B betrieben werden und würde von dort aus eine Messung zum Ziel A durchgeführt werden, könnte die fehlende Verbindung gefunden werden.

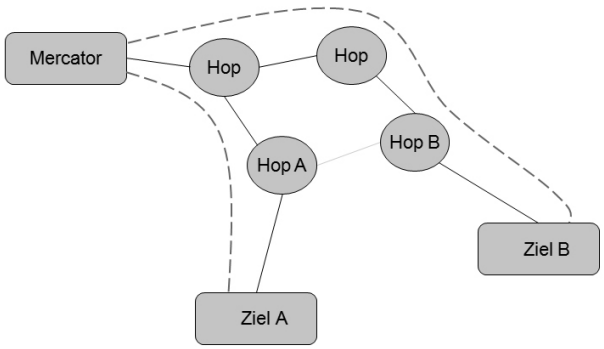


Abbildung 4: Verbindungen werden nicht immer gefunden

Um dieses Problem zu lösen, setzt Mercator auf Router, die „Source-Routing“ unterstützen. Diese Technik basiert auf dem Internet Protokoll (IP). Dabei kann ein IP Paket über einen vorgegebenen Router gesendet werden (siehe Abbildung 5). Dafür wird im IP Header des Paketes der Pfad eingefügt, den das Paket ablaufen soll. Das Paket wird dann zum ersten Knoten im angegebenen Pfad gesendet. Unterstützt dieser Knoten die Source-Routing-Technik, wird er das Paket zum

nächsten Knoten im angegebenen Pfad weiterleiten. Mercator nutzt diese Technik und führt zusätzlich zu seinen normalen Messungen, weitere Messungen über jeden bekannten Source-Routing-Router durch [9]. Bei einer Messung über einen Zwischenpunkt wählt Mercator die Initial-TTL so, dass das erste Traceroute-Paket beim Source-Routing-Router ankommt. Dann wird die TTL wieder pro Schritt um eins erhöht und somit der Pfad vom Zwischenpunkt zum Zielpunkt herausgefunden.

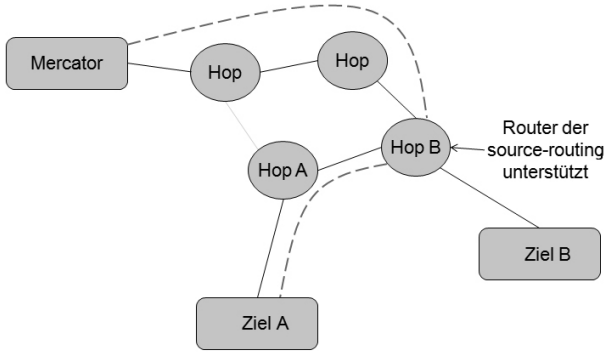


Abbildung 5: Traceroute mit Hilfe von „Source-Routing“

Source-Routing stellt jedoch ein Sicherheitsrisiko dar und ist deswegen auf den meisten Routern deaktiviert [2]. Die Entwickler haben herausgefunden, dass zum Zeitpunkt der Messungen etwa 8% aller Router im Internet Source-Routing unterstützen. Dies liegt zum Teil daran, dass Betreiber von Routern vergessen haben, Source-Routing in ihren Routern zu deaktivieren oder es nutzen, um ihr eigenes Netzwerk zu testen. Die Mercator Entwickler haben mit Simulationen berechnet, dass bei rund 5% Source-Routing-Routern bis zu 90% aller Knoten und Verbindungen in einem Netzwerk entdeckt werden können [9].

Mercator besitzt bei Programmstart noch keine Information über Router, die Source-Routing unterstützen. Daher muss es zur Laufzeit Router testen, ob bei diesen Source-Routing aktiviert ist. Hierfür sendet Mercator über einen Router A ein UDP-Paket an einen zufälligen Port eines beliebigen Routers B. Ist Source-Routing auf Router A aktiviert, leitet dieser das Paket weiter. Da das Paket an einen zufälligen Port adressiert ist, antwortet Router B sehr wahrscheinlich mit einer ICMP-port-unreachable-Fehlermeldung. Durch diese Fehlermeldung erfährt Mercator, dass Router A das Paket erfolgreich weitergeleitet hat, und Router A wird zur Liste der Source-Routing-Router hinzugefügt. Diesen Test führt Mercator mehrfach mit jedem gefundenen Router im Netzwerk aus, da möglicherweise Router A gerade nicht erreichbar ist oder Router B keine ICMP-port-unreachable-Fehlermeldung versendet, da ein existierender Port adressiert wurde.

4.3.5 Alias-Auflösung

Wie bereits in Kapitel 3.1 erwähnt, können Router mehrere Interfaces besitzen. Zur Erstellung von Internet-Karten ist es allerdings wichtig, den Router, zu dem das Interface gehört, zu kennen. Dieser Router könnte schließlich noch weitere Interfaces besitzen, die auf der Karte als ein Knotenpunkt dargestellt werden sollen.

Wie in Abbildung 6 zu sehen, führt Mercator mehrere Messungen aus. Beispielsweise könnte bei einer Messung Interface A gefunden werden und bei einer weiteren Messung Interface B. Für Mercator stellen beide Interfaces zwei eigenständige Knotenpunkte auf der Karte dar. Jedoch könnten beide Interfaces zu einem physikalischem Gerät gehören, siehe Abbildung 7. Mercator muss also in der Lage sein, mehrere Interfaces einem Router zuordnen zu können. Ansonsten würde die entstehende Internet-Karte fehlerhaft sein und das gegebene Netzwerk nicht korrekt wiedergeben werden.

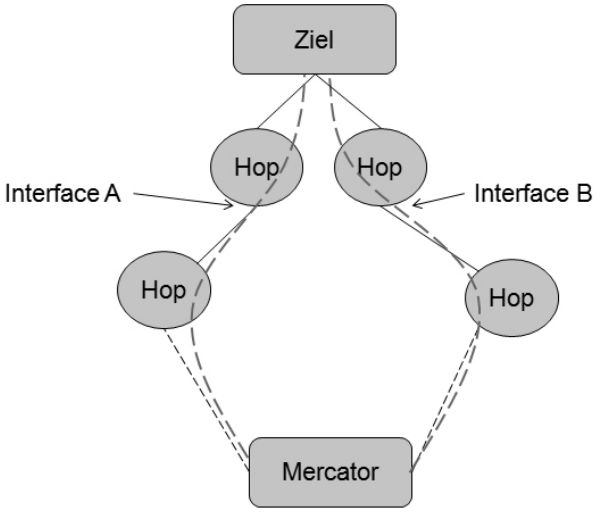


Abbildung 6: Netzwerk aus Sicht Mercators

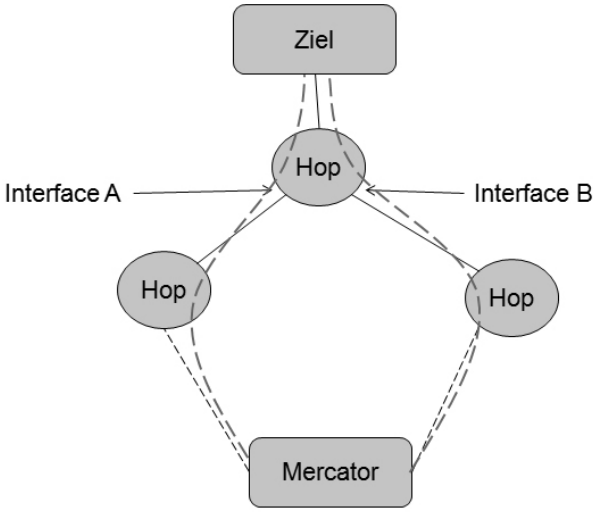


Abbildung 7: tatsächlich gegebenes Netzwerk

Um dieses Problem zu lösen, nutzt Mercator eine Eigenschaft vieler Router aus [9]. Empfängt ein Router ein Paket und möchte darauf antworten, versendet er die Antwort nicht immer über das Eingangsinterface, sondern über ein vorgegebenes Ausgangsinterface. Ein- und Ausgangsinterface sind also nicht immer gleich. Um diese Eigenschaft zu nutzen, sendet Mercator eine so genannte „alias-probe“ an ein Interface A, siehe Abbildung 8. Eine alias-probe bezeichnet ein UDP-Paket, das an einen zufälligen Port adressiert

ist. Erreicht dieses Paket den Router von Interface A, und existiert dieser Port nicht an dem Router, sendet dieser eine ICMP-port-unreachable-Fehlermeldung zurück an Mercator. Wie bereits erwähnt, kann diese Antwort möglicherweise über ein anderes Interface B versendet werden. Sendet Mercator also eine alias-probe an Adresse A und bekommt die Antwort von Adresse B zurück, so geht Mercator davon aus, dass Interface A und B zu einem physikalischem Gerät gehören [9].

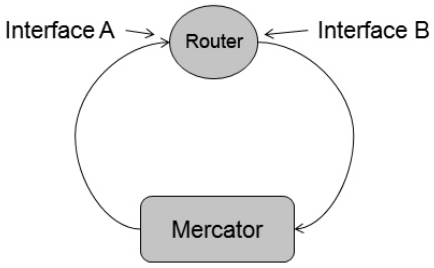


Abbildung 8: Auflösung von Alias durch „alias-probes“

Wie bei der Suche nach Source-Routing-Routern versendet Mercator wiederholt an jedes bekannte Interface eine alias-probe, um somit Alias-Interfaces aufzulösen. Jedoch kann auf Grund von vorgegebenen Netzwerkregeln und Richtlinien nicht jedes Interface direkt adressiert werden. Deswegen sendet Mercator zusätzlich noch alias-probes über Source-Routing-Router, um somit eine größere Anzahl an Interfaces zu erreichen.

4.4 Ergebnisse und Bewertung

Um Mercator zu testen, ließen die Entwickler das Programm drei Wochen lang laufen. Bei maximal 15 gleichzeitigen Traceroute-Messungen wurden knapp 150.000 Interfaces und 200.000 Verbindungen gefunden. 16% der Verbindungen wurden durch Source-Routing gefunden. 20.000 Interfaces konnten zu ihren zugehörigen Router aufgelöst werden (3.000 davon über Source-Routing-Router) [9].

Obwohl mit Traceroute-Messungen viele Interfaces gefunden werden konnten, hat Traceroute einige Probleme. So können zum Beispiel, wie in Referenz [4] erläutert, durch Load-Balancer falsche Pfade entstehen, und diese verfälschen somit die Messdaten. Weiterhin ist nicht zugesichert, dass durch Traceroute entstehende ICMP-Fehlermeldungen bei Mercator ankommen. Teilweise sind Firewalls so eingestellt, dass sie diese Meldungen verwerfen und somit der adressierte Router für Mercator als unerreichbar erscheint.

Um den Adressbereich für die Messungen einzuschränken, verwendet Mercator die Technik „informed random address probing“. Um die Effizienz dieser Technik zu messen, haben die Entwickler von Mercator die gefundenen Adressblöcke mit den Adressblöcken eines Backbone Routers verglichen. Dabei wurden nur 8% der Präfixe im Backbone Router von Mercator nicht gefunden. Wohingegen 20% Präfixe gewählt wurden, die wiederum nicht im Backbone Router vorhanden waren. Dies ist jedoch nicht verwunderlich, da durch Annahme zwei, siehe Kapitel 4.3.2, immer Nachbarblöcke hinzugenommen wurden. Diese enthalten aber möglicherweise gar keine erreichbaren Adressen.

Um die resultierende Internet-Karte zu validieren, haben die Mercator-Entwickler die Daten mit dem Netzwerk eines lokalen Internet-Service-Providers verglichen. Dabei konnten alle Router und alle Verbindungen, bis auf eine, gefunden werden [9]. Obwohl diese Ergebnisse sehr gut sind, bleibt es fraglich, ob auch in größeren, komplexeren Netzwerken ein ebenso gutes Resultat erzielt werden kann.

Mercator ist jedenfalls nicht in der Lage, von jedem Netzwerk eine komplette Internet-Karte zu erstellen. Es können Router oder Verbindungen existieren, die nicht entdeckt werden. Außerdem können Zusammenhänge, wie zum Beispiel zwei Router, die im selben Rechenzentrum platziert sind, aber nie Daten austauschen, nicht herausgefunden werden. Allerdings kann Mercator die wichtigsten Transit- und Backbone-Router finden und somit den wichtigsten Teil eines Netzwerkes darstellen.

Die resultierende Karte kann auch nicht als Momentaufnahme des Netzwerkes gesehen werden. Die Messungen vollstrecken sich über mehrere Wochen, und in dieser Zeit kann sich das Netzwerk verändern. Somit ist die Karte nur als Zeitdurchschnitt zu sehen [9].

5. VERBESSERTE METHODEN

Die Entwicklung Mercators ist bereits einige Jahre her. In dieser Zeit wurden verschiedene Techniken und Programme entwickelt, die bestehende Techniken Mercators verbessern.

Zum Standard-Traceroute-Programm, wie es auch Mercator in einer ähnlichen Form implementiert, gibt es bereits Alternativen. Paris-Traceroute ist eine Traceroute-Implementierung, die die Probleme mit Load-Balancern umgehen kann [3, 4]. Um die zunehmende Problematik mit Firewalls zu vermeiden, kann das Tool tcptraceroute [14] eingesetzt werden. Dieses verwendet anstatt ICMP-Pakete TCP Pakete. Diese sind an Port 80 adressiert, der bei fast allen Webservern geöffnet ist.

Mercators Alias-Auflösung basiert nur auf dem Fakt, dass viele Router Pakete über ein Standard-Ausgangsinterface versenden. Sollte ein Router dieser Regel nicht folgen, kann Mercator diesem Router Interfaces nicht zuordnen. Alternativ kann das Tool RadarGun [6] Alias auflösen, in dem es das Identification Feld des IP Headers betrachtet. RadarGun ist somit nicht auf das Verhalten der Router angewiesen.

6. ZUSAMMENFASSUNG

Internet-Karten bieten eine gute Möglichkeit, die Übersicht über die Struktur des Internets zu behalten. Die Daten für solche Karten können durch einfache Traceroute-Messungen generiert werden. Allerdings sind diese Karten niemals vollständig und fehlerfrei. Zusätzlich sind die Messungen sehr aufwändig und zeitintensiv, weshalb diese immer über einen längeren Zeitraum hinweg durchgeführt werden müssen.

Mercator bietet einfache Techniken, um möglichst effizient Internet-Karten zu erstellen. Bei kleinen, lokalen Teilnetzwerken gelingt dies Mercator sehr gut, ob es auch bei größeren Teilen des Internets eine relativ vollständige Karte liefert, bleibt fraglich.

Durch neuere, verbesserte Techniken könnte Mercator kor-

rektere Ergebnisse liefern. Es gibt einige aktuellere Tools, die bereits solche Techniken einsetzen. Derzeit wird auch am Lehrstuhl für Netzarchitekturen und Netzdienste der TU München eine Mess-Infrastruktur namens IStruktA entwickelt, die der Generierung von Internet-Karten dient. IStruktA bietet zu den normalen Traceroute-Messungen noch die Möglichkeit an, Messungen von beliebigen Rechnern zu starten. Somit können Internetbenutzer rund um den Globus selbstständig Messungen von ihrem Rechner aus durchführen. Dieser neuartige Ansatz verspricht zusätzliche Daten, die zu einem verbesserten Ergebnis bei Internet-Karten führen können [10].

7. LITERATUR

[1] S. S. Ands. Discovering internet topology.

[2] R. Atkinson. Security architecture for the internet protocol. Technical report, 1998.

[3] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira. Avoiding traceroute anomalies with paris traceroute. In *In Proc. Internet Measurement Conference*, 2006.

[4] B. Augustin, T. Friedman, R. Teixeira, U. Pierre, and M. Curie. Multipath tracing with paris traceroute. In *in Proc. Workshop on End-to-End Monitoring (E2EMON*, 2007.

[5] S. Bajaj, L. Breslau, D. Estrin, K. Fall, S. Floyd, P. Haldar, M. Handley, A. Helmy, J. Heidemann, P. Huang, S. Kumar, S. McCanne, R. Rejaie, P. Sharma, K. Varadhan, Y. Xu, H. Yu, and D. Zappala. Improving simulation for network research. Technical Report 99-702b, University of Southern California, March 1999. revised September 1999, to appear in IEEE Computer.

[6] A. Bender, R. Sherwood, and N. Spring. Fixing ally's growing pains with velocity modeling.

[7] CAIDA. Skitter. Online verfügbar unter http://www.caida.org/tools/measurement/skitter; besucht 24. Juni 2010.

[8] A. Danesh, , A. Danesh, L. Trajkovic, S. H. Rubin, and M. H. Smith. Mapping the internet, 2001.

[9] R. Govindan and H. Tangmunarunkit. Heuristics for internet map discovery, 2000.

[10] T. M. Lehrstuhl für Netzarchitekturen und Netzdienste. Istrukta. Online verfügbar unter http://istrukta.net.in.tum.de; besucht 24.Juni 2010.

[11] W. g. Matthew Luckie. scamper. Online verfügbar unter http://www.caida.org/tools/measurement/scamper; besucht 24.Juni 2010.

[12] G. Phillips, S. Shenker, and H. Tangmunarunkit. Scaling of multicast trees: Comments on the chuang-sirbu scaling law, 1999.

[13] N. Spring, R. Mahajan, and D. Wetherall. Measuring isp topologies with rocketfuel. In *In Proc. ACM SIGCOMM*, pages 133–145, 2002.

[14] M. C. Toren. tcptraceroute. Online verfügbar unter http://michael.toren.net/code/tcptraceroute; besucht 24. Juni 2010.

MapReduce

Dhyan Blum
Betreuer: Dirk Haage
Seminar Innovative Internettechnologien und Mobilkommunikation SS2010
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: blumd@in.tum.de

KURZFASSUNG

Die Firma Google verarbeitete im Jahr 2007 täglich 20 Petabyte an Daten [8]. Diese Leistung wäre ohne den massiven Einsatz verteilter, paralleler Systeme nicht möglich. Die Parallelisierung von Programmen ist jedoch eine komplexe Aufgabe, der zusätzliche Entwicklungsaufwand fällt zudem bei jeder weiteren verteilten Anwendung erneut an.

Diese Arbeit beschreibt das von Google entwickelte Modell *MapReduce* und dessen Implementierung. Mit *MapReduce* können extrem große Datensätze in verteilten Umgebungen verarbeitet oder generiert werden, ohne dass sich der Entwickler mit Details der Parallelisierung beschäftigen oder über Vorkenntnisse in diesem Bereich verfügen muss. Er hat lediglich die Funktionen *map* und *reduce* zu implementieren, da sich das *MapReduce*-Framework bei der Programmausführung automatisch um Parallelisierung, Lastverteilung, Fehlerbehandlung und die Kommunikation der Rechner untereinander kümmert.

Vielfältige Anwendungsmöglichkeiten ergeben sich vor allem in den Bereichen Indexierung, Tokenisierung, Suche, Erzeugung von Graphen und Datenstrukturen, maschinelles Lernen sowie *Data Mining* und Stapelverarbeitung. Bislang kaum dokumentiert ist hingegen die Verarbeitung und Analyse von Netzwerkmessdaten, was gerade deshalb verwundert, da hier mitunter sehr große Mengen an Verkehrsdaten anfallen. Heute wird das Modell *MapReduce* von vielen Entwicklern erfolgreich eingesetzt, vor allem große IT-Firmen sind in diesem Bereich stark vertreten.

1. EINLEITUNG

In den letzten beiden Jahrzehnten fand Datenverarbeitung in zunehmend größeren Dimensionen statt, gerade in Netzwerken fallen inzwischen riesige Datenmengen an. Gleichzeitig lässt sich eine Entwicklung hin zu einer immer umfassenderen Verarbeitung und Analyse dieser Daten beobachten, etliche Firmen operieren hier bereits im Petabyte-Bereich. Die Notwendigkeit, derart große Datenmengen verarbeiten zu können, wird speziell bei Suchmaschinen im Internet sehr deutlich. Um Suchanfragen möglichst schnell und umfassend bedienen zu können, müssen Milliarden von Internetseiten eingelesen, aufbereitet und analysiert werden.

Um Probleme dieser Größenordnung in annehmbarer Zeit lösen zu können, reichen einzelne Rechenmaschinen nicht mehr aus. Es müssen daher hunderte bis tausende von Computern zu einem Rechnerverbund zusammengeschlossen werden. Damit diese kooperativ arbeiten können, müssen die Eingabedaten sinnvoll verteilt und die eigentliche Berechnung parallelisiert werden. Anschließend müssen die Ergebnisse noch überprüft, zusammengeführt und unter Umständen dauerhaft gespeichert werden.

Dabei werden Entwickler vor eine ganze Reihe von Herausforderungen gestellt: Wie wird ein Problem richtig aufgeteilt? Sind überhaupt alle Probleme und ihre Berechnung parallelisierbar? Wie können die Rechner im Verbund koordiniert werden? Wie reagiert man auf Hardwareausfälle und Fehler bei der Verarbeitung? Nicht zuletzt stellt sich die Frage, wie Ergebnisse gespeichert werden, die zu groß für einzelne Computer sind.

Obwohl das ursprüngliche Problem und die nötigen Berechnungen häufig einfach sind, muss der Entwickler also eine vergleichsweise komplexe Lösung finden. Ein Großteil der Anstrengungen wird dabei in die Parallelisierung statt in die Lösung der Aufgabe investiert. Und das immer wieder, obwohl sich nur die Anwendung, nicht aber der Unterbau für die Ausführung auf verteilten Systemen ändert.

Entwickler der Firma Google haben daher im Jahr 2004 eine Lösung in Form eines Programmiermodells und einer zugehörigen Implementierung entwickelt. Sie ließen sich dabei von den in funktionalen Programmiersprachen verbreiteten *map*- und *reduce*-Funktionen inspirieren und nannten das Verfahren daher *MapReduce*. *MapReduce* kann als Framework aufgefasst werden, dessen Abstraktionsschicht alle Details der Parallelisierung vor dem Entwickler verbirgt. Dieser muss nur mehr die beiden vorgegebenen Funktionen *map* und *reduce* implementieren und kann sich somit voll auf die Lösung seines eigentlichen Problems konzentrieren.

Diese Arbeit gibt zunächst eine Einführung in die grundlegende Funktionsweise von *MapReduce*. Anschließend werden einige Beispiele vorgestellt und es wird ausführlich auf den Programmablauf eingegangen. In Abschnitt 4 folgt eine Beschreibung der Architektur und Details der Google-Implementierung von *MapReduce*. In Abschnitt 5 werden Anwendungsmöglichkeiten des Modells beschrieben, mögliche Anforderungen an Umgebung und Daten besprochen und schließlich auch die Grenzen von *MapReduce* aufgezeigt. Abschnitt 8 widmet sich verschiedenen alternativen Implementierungen des Modells. Zuletzt folgt ein Verweis auf wichtige verwandte Arbeiten und ein zusammenfassendes Fazit dieser Arbeit und ihrer Ergebnisse.

2. FUNKTIONSWEISE

Abbildung 1 veranschaulicht die grundlegende Funktionsweise von *MapReduce*. Eingabe und Ausgabe aller Berechnungen sind Mengen von `(key,value)`-Paaren. Die vom Benutzer zu implementierenden Funktionen *map* und *reduce* sind wie folgt definiert:

- *map* nimmt ein Paar `(key1,value1)` entgegen und gibt eine Liste von Paaren `list(key2,value2)` zurück. Diese Liste stellt das Zwischenergebnis dar. *MapReduce* gruppiert alle Zwischenergebnisse nach ihren Schlüsseln `key2` und reicht diese an *reduce* weiter.

- *reduce* nimmt einen Schlüssel `key2` sowie eine Liste von Werten `list(value2)` entgegen und gibt eine potentiell kleinere Liste von Werten - in der Regel nur einen oder gar keinen Wert - zurück.

3. BEISPIELE

3.1 Worthäufigkeit

Ein Problem, das sich sehr einfach für die Lösung mittels *MapReduce* formulieren lässt, ist die Ermittlung der Worthäufigkeit jedes eindeutigen Wortes in einer Menge von Dokumenten. Der Quelltext, den der Entwickler schreiben würde, sähe vereinfacht wie in Listing 1 gezeigt aus. Die *map*-Funktion gibt jedes gefundene Wort zusammen mit seiner Häufigkeit zurück, wobei die Häufigkeit in diesem einfachen Beispiel schlicht eine Eins für jedes Auftreten des Wortes ist. Die *reduce*-Funktion erhält anschließend die gruppierten Ergebnisse der *map*-Funktion, also ein einzelnes Wort und eine Liste von Einsen. Alles was die *reduce*-Funktion jetzt noch zu tun hat, ist die Einsen für jedes Wort zu summieren. Abbildung 2 illustriert den Ablauf des Programms.

```
map(String key, String value):
    // key:   document name
    // value: document contents
    for each word w in value:
        emitIntermediate(w, "1");

reduce(String key, Iterator values):
    // key: a word
    // values: a list of counts
    int result = 0;
    for each count c in values:
        result += ParseInt(c);
        emit(AsString(result));
```

Listing 1: Worthäufigkeit

3.2 Weitere Beispiele

Verteiltes Suchen

Eine *grep*-ähnliche Suchfunktion für die verteilte Suche in großen Datenmengen lässt sich ebenfalls einfach formulieren. Die *map*-Funktion gibt dabei eine Zeile zurück, wenn diese auf das Suchmuster zutreffende Zeichenketten enthält. Da weiter kein *reduce*-Vorgang nötigt ist, stellt *reduce* einfach nur die Identitätsfunktion dar und reicht die Zwischenergebnisse zur Ausgabe durch.

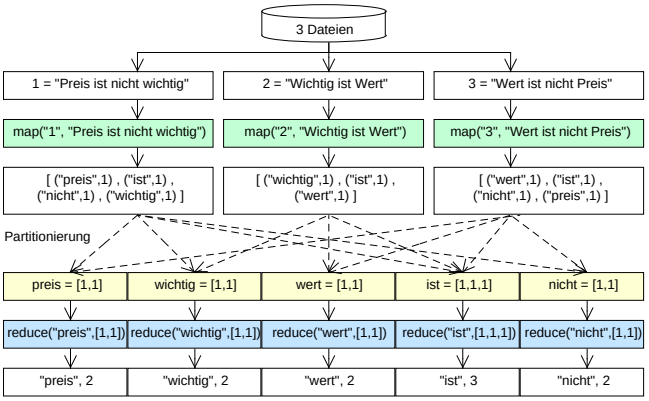


Abbildung 2: Ermittlung der Häufigkeitsverteilung von Wörtern mittels MapReduce

Zugriffsstatistiken

Um Zugriffe auf eine Menge von Internetseiten auszuwerten, lässt man die *map*-Funktion Logdateien verarbeiten und ein `(URL,"1")`-Paar für jeden Zugriff, also für jedes einzelne Vorkommen einer URL, zurückgeben. Die *reduce*-Funktion summiert alle zu einer URL gehörenden Einsen auf und gibt dann ein `(URL,totalCount)`-Paar zurück.

Verteiltes Sortieren

Beim verteilten Sortieren extrahiert die *map*-Funktion den Schlüssel eines jeden Datums und gibt ein `(key,data)`-Paar zurück. Die *reduce*-Funktion gibt die Ergebnisse erneut unverändert weiter. Die eigentliche Sortierung geschieht durch Ausnutzen der Ordnungsgarantie von *MapReduce* (vgl. Abschnitt 4.7).

4. ARCHITEKTUR

Der folgende 4. Abschnitt beschreibt die Architektur von *MapReduce*. Sofern nicht anders angegeben, ist dabei stets Googles Implementierung des Frameworks gemeint. Diese wurde in C++ geschrieben und zielt auf große Rechnerverbunde mit gewöhnlicher Hardware ab. *MapReduce* als Modell ermöglicht jedoch Implementierungen in beliebigen Sprachen und die Anpassung an unterschiedlichste Zielsysteme. Seit 2004 sind viele weitere Implementierungen veröffentlicht worden, von denen einige in Abschnitt 8 vorgestellt werden.

4.1 Google File System

Google verwendet insbesondere für seine Suchmaschine, aber auch für *MapReduce*, ein selbst entwickeltes, verteiltes Dateisystem namens *Google File System (GFS)*. GFS ist ein skalierbares System für verteilte Anwendungen und große Datenmengen. Es wurde daher mit besonderem Augenmerk auf Fehlertoleranz, Performanz und den Einsatz günstiger Standardhardware konzipiert [18]. Alternative *MapReduce* Implementierungen bauen in der Regel auf vergleichbaren Dateisystemen auf.

4.2 Vorbereitung

Bevor ein Entwickler *MapReduce* Operationen durchführen kann, muss er ein Benutzerprogramm schreiben. Dieses im-

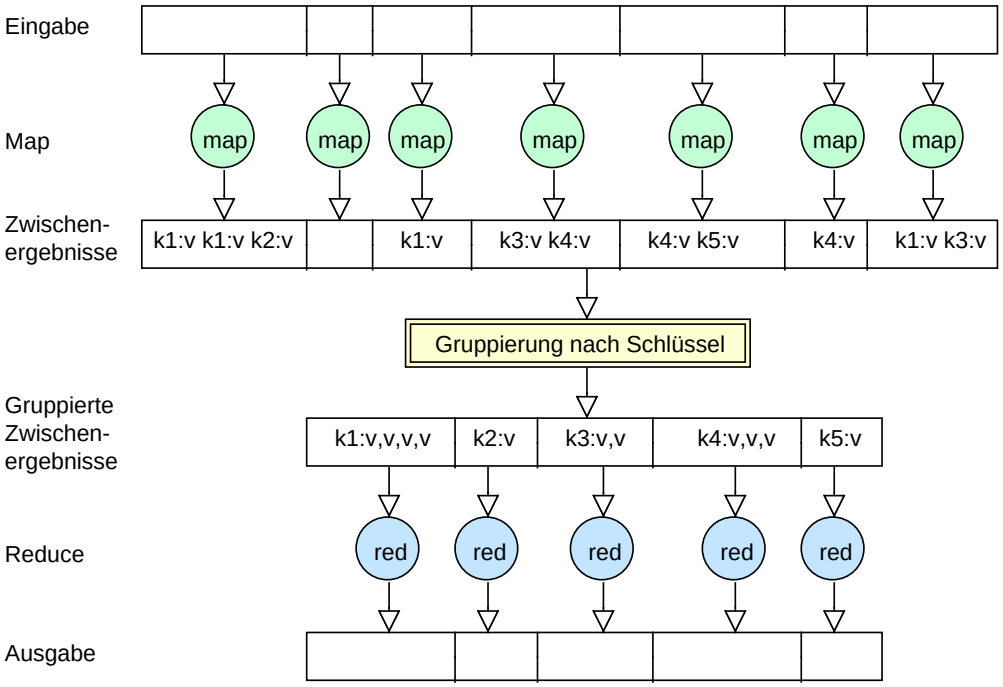


Abbildung 1: Grundlegende Funktionsweise von MapReduce

plementiert die vorgeschriebenen Funktionen *map* und *reduce* und erzeugt ein Objekt vom Typ `MapReduceSpecification`. Diesem Objekt werden die Namen der Ein- und Ausgabedateien sowie optional einige Parameter zur Optimierung der Berechnungen übergeben. Zuletzt ruft das Programm die Funktion `MapReduce()` auf und übergibt ihr das Spezifikationsobjekt. Von nun an übernimmt die *MapReduce*-Bibliothek die Kontrolle, das Benutzerprogramm wartet.

4.3 Ausführung

Abbildung 3 zeigt die einzelnen Phasen, die während der Ausführung durchlaufen und im folgenden beschrieben werden:

1. Die *MapReduce*-Bibliothek teilt die Eingabedateien automatisch in gleich große Teile auf, wobei die Stückgröße mittels optionaler Parameter vom Benutzer bestimmt werden kann. Anschließend startet die Bibliothek auf jedem angeschlossenen Rechner eine Kopie des Benutzerprogramms.

2. Eine spezielle Programmkopie ist das Masterprogramm. Genau ein Rechner erhält eine solche Kopie und wird damit zum Master, alle anderen Rechner agieren fortan als Worker. Worker, die sich im Leerlauf befinden, erhalten vom Master Aufgaben zugeteilt, die entweder aus *map*- oder *reduce*-Schritten bestehen.

3. Worker, denen eine *map*-Aufgabe zugewiesen wurde, lesen die zugehörigen Eingabedaten ein, extrahieren `(key,value)`-Paare und geben diese an die *map*-Funktion des Benutzerprogramms weiter. Die Zwischenergebnisse der *map*-Funktion werden anschließend im Arbeitsspeicher des Workers gepuffert.

4. In regelmäßigen Abständen werden die gepufferten Zwischenergebnisse gemäß einer Partitionsfunktion (vgl. Abschnitt 4.7) aufgeteilt und auf die lokale Festplatte des Workers geschrieben. Der Speicherort dieser Zwischenergebnisse wird dann dem Master mitgeteilt, der für die Weiterleitung der Information an die *reduce*-Worker verantwortlich ist.

5. Wenn ein Worker vom Master über diese Speicherorte informiert wurde, verwendet er einen *Remote Procedure Call* um die Daten von der lokalen Festplatte des *map*-Workers zu lesen. Sobald alle Zwischenergebnisse eingelesen wurden, sortiert sie der *reduce*-Worker nach ihren Schlüsseln, so dass gleiche Schlüssel gruppiert werden. Die Sortierung ist notwendig, da in der Regel viele verschiedene Schlüssel an den gleichen *reduce*-Task weitergegeben werden. Sind die Zwischenergebnisse zu groß für den Arbeitsspeicher, wird eine externe Sortierung durchgeführt.

6. Der *reduce*-Worker durchläuft nun alle sortierten Zwischenergebnisse und reicht für jeden eindeutigen Schlüssel ein Paar bestehend aus eben diesem Schlüssel und einer Liste der zugehörigen Werte an die *reduce*-Funktion weiter. Wie schon am Beispiel der Häufigkeitsverteilung von Wörtern gesehen, schrumpft die Datenmenge

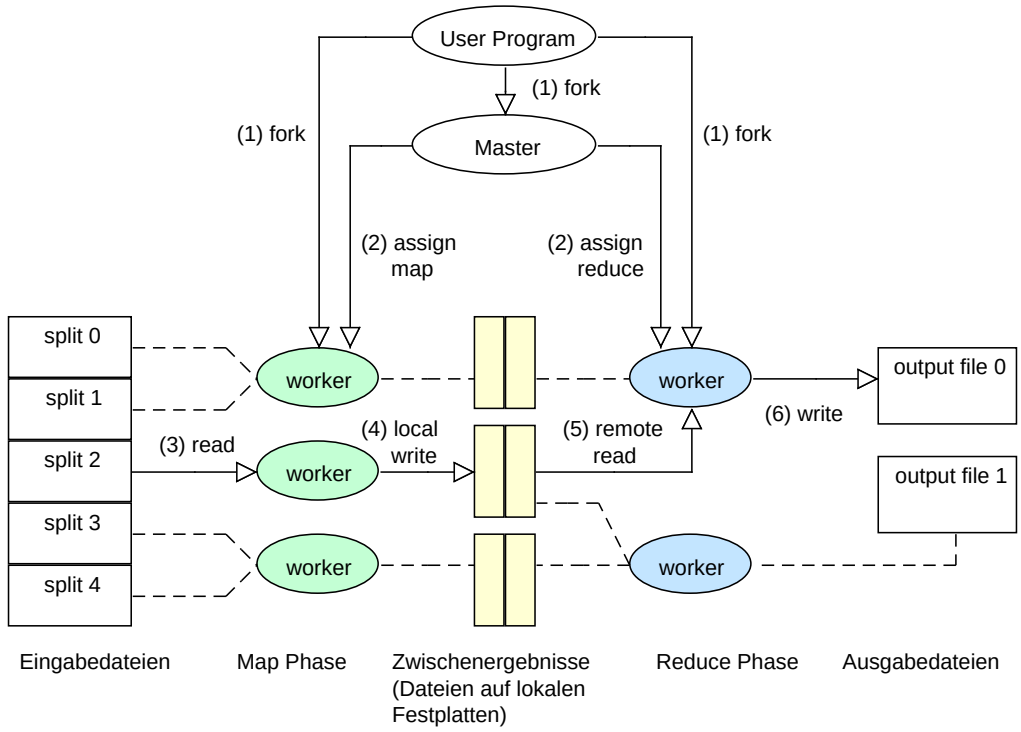


Abbildung 3: Ausführungsübersicht

nach einem Aufruf der *reduce*-Funktion in der Regel erheblich. Das Ergebnis der Funktion wird nun in eine zum aktuellen Schlüssel gehörende Ausgabedatei geschrieben. Die Namen der Ausgabedateien entsprechen der Spezifikation des Benutzers.

7. Wenn alle *map*- und *reduce*-Tasks abgeschlossen sind, weckt der Master das Benutzerprogramm wieder auf und gibt die Kontrolle an selbiges zurück. Die Ausführung wird dann nach dem Aufruf von *MapReduce* fortgesetzt.

4.4 Attribute des Masters

Der Master ist ein gewöhnlicher Rechner des Clusters, der vor der eigentlichen Ausführung eine spezielle Masterkopie des Benutzerprogramms erhalten hat. Um seine Rolle als Koordinator ausfüllen zu können, muss er eine Reihe von Datenstrukturen mit Informationen über den Zustand der aktuellen Operation verwalten. So speichert er für jeden *map*- und *reduce*-Task dessen jeweiligen Status (*idle*, *in progress* oder *completed*) sowie die Identität des zugehörigen Workers, falls der Task gerade ausgeführt wird. Außerdem muss er die Speicherorte von Zwischenergebnissen verwalten, um diese Information später entsprechenden *reduce*-Tasks zur Verfügung stellen zu können.

4.5 Fehlerbehandlung

Googles Implementierung von *MapReduce* ist auf einen Rechnerverbund zugeschnitten, der aus hunderten bis tausenden Computern mit handelsüblicher Hardware besteht. Ausfälle von Rechnern sind daher an der Tagesordnung und müssen von der Bibliothek toleriert werden. Solange die benutzerdefinierten Funktionen *map* und *reduce* deterministisch sind, garantiert *MapReduce* außerdem identische Ergebnisse wie bei vollständig sequentieller Ausführung in einer ausfallsicheren Umgebung. Um diese Eigenschaft sicherstellen zu können, wird für konkurrierende Programmpunkte auf atomare Operationen des darunter liegenden Dateisystems zurückgegriffen.

Fehler der Worker

Worker fallen häufig aus, daher muss unbedingt eine konstruktive Fehlerbehandlung stattfinden. In Googles Implementierung senden die Worker in regelmäßigen Abständen einen *ping* an den Master, um zu signalisieren, dass sie noch aktiv sind. Meldet sich ein Worker über einen festgelegten Zeitraum nicht mehr, so markiert ihn der Master als *failed* und versetzt alle vom Worker aktuell in Arbeit befindlichen und bereits verarbeiteten *map*-Tasks in den Zustand *idle*. Ein Task, der sich im Zustand *idle* befindet, kann vom Master wieder einem freien Worker zugewiesen werden. Auch die vollständig ausgeführten *map*-Tasks müssen erneut ausgeführt werden, da ihre Ergebnisse lokal auf dem Rechner des ausgefallenen Workers liegen und nicht mehr zugreifbar sind.

Fehler des Masters

Da ein Ausfall des Masters unwahrscheinlich ist, findet in Googles Implementierung keine besondere Fehlerbehandlung

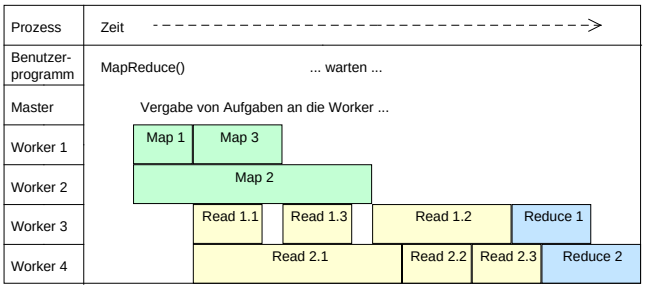


Abbildung 4: Pipelining

statt. Die gesamte Berechnung wird abgebrochen und der Benutzer hiervon in Kenntnis gesetzt. Er kann den Job dann gegebenenfalls neu starten. Eine mögliche konstruktive Fehlerbehandlung bestünde darin, den Zustand des Masters, d.h. die von ihm verwalteten Datenstrukturen, in regelmäßigen Abständen zu sichern und bei einem Ausfall einfach eine Kopie des zuletzt gesicherten Masterprogramms zu starten.

4.6 Optimierungen

Im folgenden werden einige Optimierungen beschrieben, die entweder Teil der *MapReduce*-Bibliothek sind oder vom Benutzer durch Verwendung optionaler Parameter vorgenommen werden können.

Googles verteiltes Dateisystem teilt Dateien in Blöcke auf und speichert diese redundant auf mehreren Rechnern des Clusters. Diese Tatsache kann genutzt werden, um Bandbreite im Netzwerk zu sparen. *MapReduce* versucht daher, *map*-Tasks auf Rechnern auszuführen, die eine lokale Kopie der entsprechenden Eingabedatei besitzen. Ist dies nicht möglich, wird zumindest versucht, einen Knoten mit möglichst geringer Entfernung im Netzwerk zu finden.

Ein Faktor, der die Ausführungszeit signifikant verlängern kann, sind sogenannte Nachzügler, also Rechner die überdurchschnittlich viel Zeit für die letzten verbleibenden *map*- und *reduce*-Operationen benötigen. Gründe hierfür können unter anderem defekte Festplatten mit verringerten Leseraten sein. Um das Problem zu lösen, stößt die Bibliothek gegen Ende der *MapReduce* Operation Kopien aller Tasks zur Ausführung an, die sich aktuell im Zustand *in progress* befinden. Anschließend werden die Ergebnisse all jener Task-Kopien verwendet, die zuerst fertiggestellt wurden.

Kleine Aufgabenpakete während der *map*-Phase verbessern nicht nur die Lastverteilung, sie minimieren auch die Zeit, die im Fehlerfall für die Neuausführung des *map*-Tasks benötigt wird. Abbildung 4 veranschaulicht diesen Pipelining-Effekt. Gängige Werte liegen bei Google etwa im Bereich von 200.000 *map*- sowie 5.000 *reduce*-Tasks auf einem Verbund von 2.000 Rechnern.

4.7 Erweiterungen

Standardmäßig werden die Zwischenergebnisse anhand ihrer Schlüssel in R Teile zerlegt und auf R *reduce*-Tasks verteilt. Der Benutzer kann jedoch eine eigene Partitionsfunktion für den Fall angeben, dass er zuvor eine Funktion auf die Schlüssel der Zwischenergebnisse anwenden möchte. Werden beispielsweise URLs als Schlüssel verwendet und man möchte statt einer Ausgabedatei pro URL lieber eine Datei pro Host, so kann durch die Partitionsfunktion zunächst der Hostname aus der URL extrahiert und anschließend als Schlüssel verwendet werden. *MapReduce* garantiert weiterhin, dass innerhalb einer Partition alle `(key,value)`-Paare aufsteigend nach ihren Schlüsseln verarbeitet werden. Diese Eigenschaft ist unter anderem beim Sortieren nützlich.

In bestimmten Fällen, zum Beispiel bei der Häufigkeitsverteilung von Wörtern, tritt der Schlüssel in den Ergebnispaaren der *map*-Funktion immer wieder auf. So würden im Deutschen unzählige `("und",1)`-Paare in die Ausgabedatei geschrieben und später über das Netzwerk übertragen. In solchen Fällen kann der Benutzer eine *combine*-Funktion bereitstellen, die auf die Ergebnisse der *map*-Funktion angewendet wird, bevor deren Zwischenergebnisse in lokale Dateien geschrieben werden. In der Regel erfüllen *combine* und *reduce* die gleiche Aufgabe. Der Unterschied liegt darin, dass die Ergebnisse von *reduce* in finale Ausgabedateien geschrieben, die von *combine* jedoch wie Zwischenergebnisse von *map* behandelt werden.

Weiterhin existieren einige vordefinierte Funktionen zur Verarbeitung verschiedener Eingabeformate, Mechanismen um unerwünschte oder fehlerhafte Datensätze zu überspringen, eine sequentielle Version von *MapReduce* für die lokale Ausführung und lokales *Debugging* sowie eine *counter*-Schnittstelle, durch die der Entwickler das Auftreten bestimmter Ereignisse während der Berechnung global zählen kann. Letzteres kann beispielsweise von Nutzen sein, um die Einhaltung bestimmter Toleranzen zu überwachen. Darüber hinaus beinhaltet Googles Implementierung von *MapReduce* auch einen internen HTTP-Server, der während der gesamten Operation Statusinformationen im Browser zur Verfügung stellt.

5. ANWENDUNGEN

Grundsätzlich ist *MapReduce* für nebenläufige Berechnungen über große Datenmengen auf Rechnerverbunden entwickelt worden. Die meisten Anwendungen fallen dabei in eine der folgenden drei Kategorien:

1. Tokenisierung, Indexierung und Suche
2. Erzeugung von Datenstrukturen wie z.B. Graphen
3. Data Mining und maschinelles Lernen

Dean und Ghemawat nennen in ihrer Arbeit [6] und den zugehörigen Vortragsfolien [7] zu *MapReduce* einige Anwendungsbeispiele: verteiltes Sortieren, verteiltes Suchen, die Erstellung von gewöhnlichen und umgedrehten Weblinkgraphen, *Term Vector Per Host*-Berechnungen (ein Teilgebiet bei der Erstellung von Suchmaschinen-*Rankings*), die Auswertung von Zugriffsstatistiken zu Internetseiten, invertierte Indexerstellung, *Document Clustering*, maschinelles Lernen sowie *Statistical Machine Translation*. Googles wichtigste Anwendung ist dabei zweifellos die Erstellung des Suchindexes, für die allein 24 verschiedene, hintereinander ausgeführte *MapReduce*-Operationen nötig sind. Michael Kleber erwähnt in [13] zudem Berechnungen im Zusammenhang mit Googles *PageRank*.

Auf der Projektseite des alternativen *MapReduce*-Frameworks *Hadoop* (vgl. Abschnitt 8, Implementierungen) findet sich eine Liste von Nutzern und deren *Hadoop*-Anwendungen. Die meisten Anwendungen gehören dabei zu einer der oben genannten Kategorien, am häufigsten fallen die Begriffe Web, Suche, Analyse, Statistik, Werbung, Indexierung, Graphen und Bildverarbeitung. Vieles davon steht im Zusammenhang mit Benutzern, deren Verhalten auf Webseiten und den dabei anfallenden Daten. Unter den Nutzern befinden sich etliche bekannte Unternehmen, so z.B. Amazon, Adobe, AOL, Baidu, Facebook, IBM, Microsoft und viele weitere. Die New York Times nutzte *Hadoop* 2007 beispielsweise, um 11 Millionen eingescannte Zeitungsartikel — 4 Terabyte — aus dem Zeitraum von 1851 bis 1980 in PDF-Dateien zu konvertieren. Die Operation wurde auf 100 *Amazon Elastic Compute Cloud* Instanzen durchgeführt und nahm 24 Stunden in Anspruch [10].

Die Verwendung von *MapReduce* im Zusammenhang mit Netzwerkmessdaten ist derzeit kaum dokumentiert, obwohl gerade hier große Datenmengen anfallen können und durchaus entsprechende Analyseprogramme existieren, die das Sammeln und Auswerten dieser Daten in verteilten Umgebungen unterstützen. Münz und Carle beschreiben in [15] die verteilte Netzwerkanalyse unter Verwendung der Programme *TOPAS* und *Wireshark*. Weiterhin haben Schneider et al. in [19] gezeigt, dass *packet capturing* mittels Standardhardware selbst in voll ausgelasteten 10-Gigabit-Netzwerkumgebungen möglich ist, sofern ein verteiltes System zum Einsatz kommt und der Datenverkehr auf mehrere einzelne Systeme verteilt wird. Gerade der Einsatz von Standardhardware wird jedoch seitens der *MapReduce*-Entwickler sehr hervorgehoben, da hier das Verhältnis von Nutzen zu Kosten besser sei, als beim Einsatz hochverfügbarer Spezialhardware [5].

Zumindest Kang et al. beschreiben in einem Lehrgang [20] den erfolgreichen Einsatz von *Hadoop* bei der Analyse von Netzwerkflussdaten. Primäres Ziel dabei sei, die benötigte Rechenzeit zu senken und die Fehlertoleranz der Analyse zu erhöhen. Die verwendete Hardwarearchitektur besteht aus mehreren Netzwerken, deren Router jeweils Flussdaten an Computer eines Rechnerverbunds übertragen. Auf der Softwareseite steht eine mehrschichtige Architektur, die in Abbildung 5 veranschaulicht wird. Aufbauend auf *Hadoop* kommen das verteilte Dateisystem *HDFS* und eine *MapReduce*-Bibliothek mit den obligatorischen *map/reduce*-Funktionen zum Einsatz. Eingehende Flussdaten werden im Binärformat gespeichert, von einem Flussdatenkonverter in ein Textformat umgewandelt und anschließend im *HDFS* abgelegt. Die *map*-Tasks lesen Daten zeilenweise ein, extrahieren den Zielport und die Länge der Daten und geben anschließend ein `(port,size)`-Paar zurück. Die *reduce*-Tasks lesen die nach Ports gruppierten Zwischenergebnisse ein und summieren die zugehörigen Datenlängen. Bei der Evaluierung vergleichen die Autoren die benötigte Rechenzeit ihrer Lösung mit der konventioneller Analyseprogramme und erreichen dabei eine um 72 Prozent kürzere Rechenzeit. Die Entwickler kommen außerdem zu dem Schluss, dass eine direkte Unterstützung binärer Eingabeformate seitens der Bibliothek die benötigte Rechenzeit noch weiter senken könnte und wollen daher ein entsprechendes Eingabemodul für *Hadoop* entwickeln.

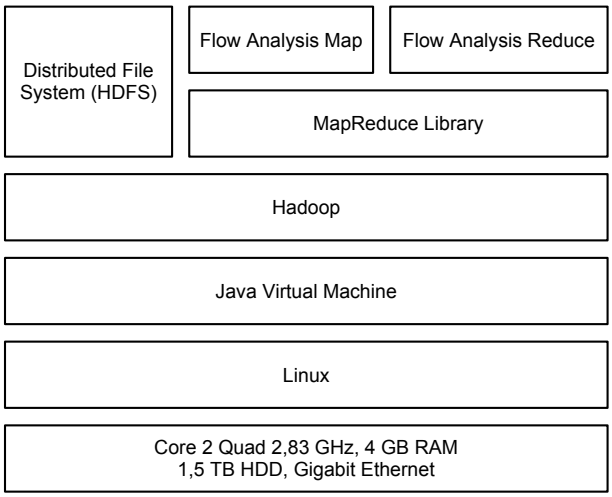


Abbildung 5: Hadoop Architektur für die Analyse von Netzflussdaten

6. ANFORDERUNGEN

MapReduce stellt keine besonderen Anforderungen an die verwendete Hardware oder die zu verarbeitenden Daten. Googles Implementierung in C++ arbeitet zwar ausschließlich mit Strings, der Konvertierung zwischen Strings und den tatsächlich benötigten Datentypen steht jedoch nichts im Wege. Auch ist der Entwickler nicht auf Dateien als Datenquelle angewiesen, durch die Implementierungen einer einfachen *reader*-Schnittstelle können Eingabedaten auch aus einer Datenbank oder aus Datenstrukturen im Speicher gelesen werden. Gleiches gilt für das Ausgabeformat. Im Zusammenhang mit Datentypen wäre lediglich zu erwähnen, dass die Ergebnisse der *map*-Funktion der gleichen Domäne wie die Ein- und Ausgaben der *reduce*-Funktion entstammen. Auch wenn die Schlüssel für das endgültige Ergebnis bedeutungslos sind, muss diese Beziehung bestehen bleiben.

7. EINSCHRÄNKUNGEN

Googles Entwickler stellten *MapReduce* 2004 auf einer Konferenz in San Francisco vor. Im Anschluss an den Vortrag wurde aus dem Publikum die Frage gestellt, welche Probleme nicht mit *MapReduce* zu lösen seien. Die Antwort darauf lautete, dass „join operations“ nicht durchgeführt werden können [17]. Michael Kleber gibt in [12] das Denkspiel *Boggle* als anschauliches Beispiel für diese Einschränkung an. Bei *Boggle* müssen Spieler in einem rechteckigen Feld mit Buchstaben möglichst viele und möglichst lange Wörter finden, wobei die Buchstaben benachbart aber nicht zwingend in einer Linie angeordnet sein müssen. Man stelle sich nun ein Spielfeld vor, das zu groß ist, um vollständig in den Speicher geladen zu werden. Um das Problem mit Hilfe von MapReduce lösen zu können, teilt man das Spielfeld in mehrere kleinere Felder auf und lässt diese durchsuchen. Wie findet man dann aber Wörter, die über die Grenzen der kleineren Felder hinausgehen? Und wie stellt man sicher, sie nur einmal zu zählen? Probleme dieser Art sind mit MapReduce nicht lösbar, da die Teilschritte während der *map*-Phase nicht unabhängig voneinander durchgeführt werden können.

8. IMPLEMENTIERUNGEN

Die bisherigen Abschnitte bezogen sich in erster Linie auf Googles Implementierung von *MapReduce*. Im folgenden werden nun einige alternative Implementierungen des *MapReduce*-Modells vorgestellt. Mit *Dryad* wird zudem eine konzeptionelle Alternative präsentiert, die zwar ähnliche Ziele, jedoch auch neuartige Ansätze verfolgt.

Das von der *Apache Software Foundation* betreute Open-Source-Framework *Hadoop* bietet unter den Namen *Hadoop MapReduce* und *Hadoop Distributed File System (HDFS)* vergleichbare Java-Implementierungen zu *MapReduce* und Googles verteiltem Dateisystem *GFS* an. Größter Unterstützer und einer der vielen namhaften Anwender von *Hadoop* ist die Firma Yahoo. Yahoo führt nach eigenen Angaben *Hadoop*-Operationen vor allem in den Bereichen Werbesysteme und Internetsuche auf mehr als 36.000 Rechnern mit 100.000 Rechenkernen durch [1].

Yahoo Pig baut auf *Hadoop* auf, bietet jedoch mit *Pig Latin* eine neuartige Abfragesprache, die die Lücke zwischen dem deklarativen Stil von SQL und dem als *low-level* empfundenen, prozeduralen Stil von *MapReduce* füllen soll. In [16] geben die *Pig*-Autoren einige Beispiele dafür an, wie stark *Pig* die benötigte Entwicklungs- und Ausführungszeit gegenüber der direkten Nutzung von *Hadoop* senken kann.

Skynet ist eine in Ruby geschriebene Implementierung des *MapReduce*-Frameworks, die als adaptives, fehlertolerantes und vollständig verteiltes System ohne *single point of failure* beworben wird. Die Open-Source-Implementierung kommt völlig ohne Master aus, stattdessen überwachen sich die Worker gegenseitig und übernehmen bei Bedarf den aktuellen Task eines anderen, ausgefallenen Workers. Zudem kann jeder Worker zu jeder Zeit in die Rolle eines koordinierenden Masters schlüpfen und wird auch in diesem Zustand von anderen Workern überwacht und gegebenenfalls ersetzt [3].

Disco ist eine von Nokia entwickelte und in Python geschriebene Implementierung von *MapReduce*. Auch *Disco* baut auf einem eigenen Dateisystem auf, dem *Disco Distributed Filesystem (DDFS)*. Mit *Discodex* wird zudem eine verteilte, auf `(key,value)`-Paaren basierte Datenbank mit entsprechenden Abfragemöglichkeiten angeboten [2].

Dryad ist eine von Microsoft entwickelte, allgemeinere Variante von *MapReduce*. *Dryad*-Anwendungen werden als gerichtete, azyklische Graphen modelliert. Der Graph bestimmt dabei den Datenfluss, die Knoten bestimmen die Operationen auf den Daten. Knotenprogramme werden streng sequentiell geschrieben und zur Ausführung automatisch auf mehrere Rechenmaschinen oder Rechenkerne verteilt. *Dryad* wurde als Plattform entwickelt, auf der weitere Schnittstellen aufbauen können, z.B. in Form von Abfragesprachen. Ein Beispiel hierfür ist die eigens entwickelte *nebula scripting language*, die einfachere, wenn auch stärker limitierte Operationen auf großen Datenmengen ermöglicht. Eine Erweiterung für den *Microsoft SQL Server* ermöglicht es zudem, SQL-Operationen im Kontext von *Dryad* durchzuführen. *Dryad* verfolgt im Grunde die gleichen Ziele wie MapReduce, soll es dem Entwickler also möglichst einfach machen, möglichst beliebige Anwendungen in verteilten Umgebungen auszuführen, ohne sich dabei um die Parallelisierung kümmern zu müssen. Die *Dryad*-Entwickler opfern jedoch nach eigenen Angaben ein Stück weit die Einfachheit von *MapReduce*, um den Entwickler im Gegenzug vom als zu starr empfundenen *map/sort/reduce*-Paradigma zu befreien [11].

9. PERFORMANZ

Google hat die Performanz von *MapReduce* in vielen experimentellen und realen Anwendungen untersucht und mitunter beeindruckende Ergebnisse erzielt. So wurde beispielsweise 2008 ein neuer Rekord im Benchmark *Terasort* aufgestellt, bei dem Daten im Umfang von einem Terabyte möglichst schnell sortiert werden müssen. Google gelang diese Aufgabe mit *MapReduce* in 68 Sekunden, während der bisherige Rekordhalter Yahoo mit 209 Sekunden dreimal langsamer war. Yahoo setzte mit *Hadoop* ebenfalls auf eine *MapReduce*-Implementierung, beide Seiten hatten etwa 1000 Server im Einsatz. Fairerweise sei jedoch gesagt, dass Google dreimal mehr Festplatten in seinen Rechnern einsetzte, was bei E/A-intensiven Operationen wie dem Sortieren entsprechende Auswirkungen hat. Die Google Entwickler gingen jedoch noch einen Schritt weiter und sortierten mit 4000 Rechnern einen Petabyte an Daten, also 1.000 Terabyte. Der Vorgang nahm sechs Stunden in Anspruch, das sortierte Ergebnis wurden auf 48.000 Festplatten gespeichert [4]. Für ausführliche Leistungsanalysen sei an dieser Stelle jedoch auf [6] verwiesen.

10. VERWANDTE ARBEITEN

Grundlage dieses Dokuments ist die 2004 erschienene Arbeit *MapReduce: Simplified Data Processing on Large Clusters* [6] sowie die zugehörigen Vortragsfolien [7] der beiden Google Entwickler Jeffrey Dean und Sanjay Ghemawat. Dean beschreibt 2006 in einem weiteren Vortrag [5] die bisher bei Google gemachten Erfahrungen mit *MapReduce*. Microsoft Entwickler Ralf Lämmel zerlegt in seiner grundlegenden Untersuchung [14] das *MapReduce*-Programmiermodell in sämtliche Einzelteile, um es dann Stück für Stück und mit starkem Bezug zu den funktionalen Sprachen wieder zusammenzusetzen.

11. ZUSAMMENFASSUNG

Diese Arbeit hat Googles *MapReduce*-Framework vorgestellt, das Programme automatisch parallelisiert und so die einfache Verarbeitung großer Datenmengen auf verteilten Systemen ermöglicht. Ausführlich wurde auf Implementierung, Programmablauf und mögliche Anwendungsgebiete eingegangen, zudem alternative Frameworks vorgestellt.

Da sich die *MapReduce*-Bibliothek um alle Details der Parallelisierung, Fehlerbehandlung, Lastverteilung und Kommunikation der Rechner untereinander kümmert, kann sie helfen den Aufwand bei der Entwicklung verteilter Anwendungen zu reduzieren. Weiterhin wurde gezeigt, dass es in bestimmten Bereichen vielfältige Anwendungsmöglichkeiten für *MapReduce* gibt. Zumindest bei der Verarbeitung und Analyse von Netzwerkmessdaten scheint das Framework jedoch noch kaum verbreitet zu sein.

Kritiker monieren, die Nutzungsmöglichkeiten seien trotz allem zu begrenzt, das Programmiermodell sei unnötig einschränkend, rückwärts gewandt und auf zu niedriger Ebene

angesiedelt. Darüber hinaus würden die *MapReduce*-Entwickler die Erfahrungen aus 40 Jahren Datenbankentwicklung ignorieren [9]. Diese Kritik mag nicht völlig unzutreffend sein, die Erfolge von *MapReduce* sind jedoch nicht zu übersehen. Erstmals wurde die Idee eines Frameworks für verteilte Anwendungen und der effiziente Ansatz zur Fehlerbehandlung durch Neuausführung einem breiten Publikum zugänglich gemacht. Nicht zu unterschätzen ist außerdem der dadurch ermöglichte Einsatz kostengünstiger Standardhardware in großen, verteilten Systemen. Vor allem aber hat *MapReduce* neben etlichen Implementierungen auch eine ganze Reihe von Weiterentwicklungen des ursprünglichen Konzepts hervorgebracht. Einige hiervon nähern sich stark dem Datenbankbereich an und können so gewisse Kritikpunkte an *MapReduce* entkräften. Andere, wie beispielsweise Microsofts *Dryad*, bringen neuartige Ansätze und bei Bedarf höhere Abstraktionsschichten für den Entwickler mit.

MapReduce war letztlich auch deshalb ein Erfolg, weil es verteilte Anwendungen bei Entwicklern populärer gemacht und ein Stück weit *demokratisiert* hat. Auf Grund der vielen aktuellen Entwicklungen in diesem Bereich, darf man gespannt sein, wie es in Zukunft weitergeht.

12. LITERATUR

[1] Apache hadoop project website. http://hadoop.apache.org/, June 2010.

[2] Disco project website. http://discoproject.org/, June 2010.

[3] Skynet project website. http://skynet.rubyforge.org/, June 2010.

[4] G. Czajkowski. Sorting 1 pb with mapreduce. http://googleblog.blogspot.com/2008/11/sorting-1pb-with-mapreduce.html, November 2008.

[5] J. Dean. Experiences with mapreduce, an abstraction for large-scale computation. Proc. 15th International Conference on Parallel Architectures and Compilation Techniques, 2006.

[6] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI '04*, pages 137–150, December 2004.

[7] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. Slides from http://labs.google.com/papers/mapreduce.html, December 2004.

[8] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, 2008.

[9] D. J. DeWitt and M. Stonebraker. Mapreduce: A major step backwards. http://databasecolumn.vertica.com/database-innovation/mapreduce-a-major-step-backwards/, January 2008.

[10] D. Gottfrid. Self-service, prorated super computing fun! http://open.blogs.nytimes.com/2007/11/01/self-service-prorated-super-computing-fun/, November 2007.

[11] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. Dryad: distributed data-parallel programs from sequential building blocks. In *EuroSys '07: Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*, pages 59–72, New York, NY, USA, 2007. ACM.

[12] M. Kleber. Mapreduce nature: Not everything is a nail. Slides from http://sites.google.com/site/mriap2008/lectures, January 2008.

[13] M. Kleber. What is mapreduce? Slides from http://sites.google.com/site/mriap2008/lectures, January 2008.

[14] R. Lämmel. Google's mapreduce programming model — revisited. *Sci. Comput. Program.*, 68(3):208–237, 2007.

[15] G. Münz and G. Carle. Distributed network analysis using topas and wireshark. In *Proceedings of IEEE Workshop on End-to-End Monitoring Techniques and Services (E2EMon 2008)*, April 2008.

[16] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins. Pig latin: a not-so-foreign language for data processing. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1099–1110, New York, NY, USA, 2008. ACM.

[17] OSDI. Conference reports. http://www.usenix.org/publications/login/2005-04/openpdfs/osdi04.pdf, December 2004.

[18] H. G. Sanjay Ghemawat and S.-T. Leung. The google file system. 19th ACM Symposium on Operating Systems Principles, October 2003.

[19] F. Schneider, J. Wallerich, and A. Feldmann. Packet capture in 10-gigabit ethernet environments using contemporary commodity hardware. In *Proceedings of the 8th International Conference on Passive and Active Network Measurement*, volume 4427 of *Lecture Notes in Computer Science*, pages 207–217, New York, NY, USA, Apr. 2007. Springer-Verlag Berlin Heidelberg.

[20] Y. L. Wonchul Kang, Yeonhee Lee. Netflow analysis with mapreduce. 3rd CAIDA-WIDE-CASFI Joint Measurement Workshop, April 2010.

Van Jacobsons Content Centric Networks

Christoph Schindlbeck
Betreuer: Dipl. Inform. Holger Kinkelin, Dipl. Inform. Marc Fouquet
Seminar Innovative Internettechnologien und Mobilkommunikation SS2010
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: schindlc@in.tum.de

KURZFASSUNG

Der Datenverkehr in Netzwerken wandelt sich immer mehr in eine Richtung, die die Daten selbst in der Vordergrund stellt. Dies steht im Gegensatz dazu, dass Informationen nach wie vor nicht über ihren Inhalt, sondern über ihren Speicherort angesprochen werden. Content-Centric Networks (CCN) machen sich genau dies zur Aufgabe und wollen über die „Adressierung“ von Daten anstatt Speicherorten in IP-Netzen diverse Probleme beseitigen. So sollen neben dem Beheben des Adressierungsproblems auch die verfügbaren Bandbreiten besser genutzt und die Sicherheit durch konsequenze Signierung erhöht werden und langfristig ein Ersatz des IP-Protokolls ermöglicht werden.

Schlüsselworte

CCN, Jacobson, Netzwerktechnik, Sicherheit

1. EINLEITUNG

Das Internet in seiner heutigen Form basiert auf Funktionsprinzipien, die in den 60er und 70er Jahren des letzten Jahrhunderts entstanden sind: Damals dienten Netzwerke dazu, um begrenzt vorhanden Ressourcen wie Bandspeichergeräte oder die Rechenleistung von Computern auf viele Nutzer zu verteilen. Daraus ist das Modell der IP-Kommunikation entstanden, das genau zwei Teilnehmer mit individuellen Kennzeichnungen (Adressen) auf einem weitgehend statischen Weg verbindet.[1] Einer der beiden will eine Ressource nutzen und fordert diese beim anderen an. Auf diesem Modell beruht bis heute nahezu der gesamte Datenverkehr im Internet.

Doch das Nutzungsverhalten von Computernetzen hat sich seitdem grundlegend verändert. Durch den starken Preisverfall und die damit verbundene massenhafte Verbreitung von technischen Geräten ist die Aufgabe von Netzwerken heute weniger das Bereitstellen von physischen Kapazitäten, als das Anbieten von bestimmten Inhalten. Diese Daten werden von einer ständig wachsenden Zahl von unterschiedlichen Geräten angeboten, so dass deren Organisation einen immer größeren Aufwand bedeutet. Wenn ein Internetnutzer zum Beispiel ein Video auf Youtube ansehen will, interessiert er sich nicht dafür, wo dieses gespeichert ist. Doch um an die gewünschten Daten zu kommen, muss der Anfragende dazu wissen, *wo* er danach suchen muss, anstatt nach dem *Inhalt selbst* zu fragen.

Daraus ergeben sich mehrere kritische Punkte, die im Umgang mit Daten beachtet werden müssen:

- **Verfügbarkeit:** Um einen schnellen, zuverlässigen Datenverkehr sicherzustellen, werden derzeit spezialisierte, den anwendungsspezifischen Gegebenheiten angepasste Infrastrukturen wie Peer-to-Peer-Netze oder Content Delivery Networks (z.B. Akamai) genutzt.
- **Sicherheit:** Die Datenintegrität wird bisher in erster Linie danach beurteilt, wo die Daten herkommen und wie die Verbindung zum Sender gesichert ist.
- **Ortsabhängigkeit:** Daten liegen an festen Stellen. Um auf sie zugreifen zu können, muss zuerst ein Zusammenhang zwischen den Inhalten und ihrem Speicherort gezogen werden (in der Regel durch das Domain Name System DNS), was viele Ressourcen verschlingt.

Dazu kommt, dass viele Datenpakete, wie etwas beliebte Videos auf Youtube, große Downloads oder über das Internet verteilte TV-Sendungen, die alle nur auf wenigen Servern vorgehalten werden, vielfach die selben Wege durch das Netz zurücklegen. Dadurch werden bei jeder Nutzung die Server des Hauptverteilers beansprucht und zudem fließt der Datenverkehr vielfach auf den gleichen Wegen.

Um diese Probleme anzugehen, hat ein Team um den Netzwerkspezialisten Van Jacobson am Palo Alto Research Center ein völlig neues Konzept der Vernetzung von Rechnern entwickelt: Die Content Centric Networks (CCN)[6]. Hierbei ist der Ansatz, Daten nicht nach ihrem Speicherort zu adressieren, sondern ihnen Namen zu geben, die sich dann ansprechen lassen.

Das CCN-Protokoll ist dabei nicht ein weiteres Protokoll, das auf dem Internetprotokoll aufbaut, sondern soll dieses weitgehend ersetzen. Um dies möglich zu machen, hat Jacobson bei der Entwicklung einen ähnlichen Ansatz verfolgt, wie beim Entwurf des IP. So soll gewährleistet werden, dass beide Protokolle koexistieren können und ein Umstieg fließend erfolgen kann. Im Gegensatz zum relativ statischen IP kann ein CCN aber sehr leicht von der Verwendung mehrerer Übertragungswege profitieren.

[1] Loadbalancer und andere Faktoren können zwar den Weg von Paketen beeinflussen, aber häufig nehmen Pakete, die zum selben Datenstrom gehören, denselben Weg. Zudem kommt es in der Regel nicht zu parallelen Verbindungen zwischen zwei Kommunikationspartnern.

Im folgenden soll zunächst der grundlegende technische Aufbau von CCNs erläutert werden, mit der Protokollstruktur sowie Erklärungen zum Datentransport und Routingprotokollen. Anschließend folgt ein Abschnitt zur Datensicherheit in CCNs. Zuletzt soll das Konzept noch kritisch betrachtet und Probleme des Entwurfs festgestellt sowie die Aussichten auf Erfolg betrachtet werden.

2. DAS CCN-PROTOKOLL

Die gesamte Kommunikation in einem CCN besteht aus zwei verschiedenen Typen von Paketen: Zum einen gibt es Interest-Pakete, die signalisieren, dass ein Teilnehmer bestimmte Daten sucht. Zudem gibt es noch die eigentlichen Datenpakete, die zum jeweiligen Interest passen. Diese Daten müssen nicht im Vorhinein existieren, sondern können auch dynamisch nach Anfrage erzeugt werden, wie etwa Webseiten mit veränderlichen Inhalten. Wenn ein Datenpaket verschickt wird, zehrt es alle zugehörigen Interest-Pakete auf dem Weg zum Anfragenden auf. Der Aufbau der Pakete wird in Abbildung 1 skizziert.

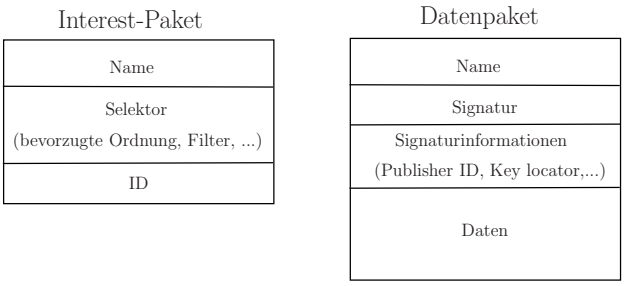


Abbildung 1: Aufbau der Pakete

2.1 Namen

Beide Paketarten sind durch einen gleichlautenden Namen eindeutig festgelegt. Die Erzeugung dieser Namen ist ein wesentliches Bestandteil des Konzepts. Wie beim IP (Netz, Subnetz, Host) basiert es auf hierarchischen Strukturen. Dadurch lassen sich die Inhalte leicht in Form von Baumstrukturen verwalten. Ein Beispiel hierfür findet sich in Abbildung 2.

Suchen können daher schnell und effizient durch ein Vergleichen der Namenspräfixe realisiert werden. Ist der angeforderte Inhalt noch nicht verfügbar, so kann er auch erst nach Empfangen eines Interest-Pakets generiert werden. Zudem können die Namen auch vom Kontext abhängen. Beispielsweise könnte der Name `/Nachrichten/heute` die aktuellen Nachrichten liefern.

Die vom CCN-Protokoll verwendeten Namen müssen keinerlei Formansprüchen genügen. Insbesondere müssen sie weder menschenlesbar sein, noch irgendeine Form auf Protokollebene besitzen. Auch die Länge der Namensteile spielt keine Rolle, genauso wie die Hierarchieebenen. Es ist lediglich wichtig, dass die jeweiligen Präfixe soweit bekannt sind, dass Interests an die richtigen Stellen weitergeleitet werden können. Die Form muss lediglich so gewählt sein, dass Protokolle auf höherer Ebene sie verstehen können. Dies ist ein entscheidender Beitrag zur Datensicherheit (siehe Kapitel 4).

2.2 Verwalten von Anfragen

Das grundlegende Funktionsprinzip ähnelt dem des bekannten Internetprotokolls: Pakete, die über eine Schnittstelle eingehen, werden analysiert und anschließend wird, abhängig vom Ergebnis der Analyse, eine Aktion ausgeführt. Es gibt dabei folgende drei Strukturen:

- Forwarding Information Base (FIB)
- Inhaltsspeicher (ein Puffer)
- Pending Interest Table (PIT)

Kann ein Interest nicht direkt beantwortet werden, so enthält die FIB die bekannten Quellen, die möglicherweise eine passende Antwort bereitstellen können. Dies erfolgt ähnlich wie beim IP, allerdings mit dem Unterschied, dass die ausgehenden Anfragen über verschiedene Netzwerkschnittstellen erfolgen können. Dadurch kann ein CCN von mehreren Verbindungen profitieren.

Der Puffer verfolgt eine andere Strategie wie beim IP: Werden die Datenpakete dort nur so lange gespeichert, bis sie weiter verschickt werden, so passiert in einem CCN das Gegenteil: Sämtliche Datenpakete bleiben auch nach dem Verschicken an den Interessenten weiter gespeichert. Erst wenn der Speicher voll ist, werden die Pakete gelöscht, die am längsten nicht mehr nachgefragt wurden (Least Recently Used-Ersetzung, LRU).

In der PIT werden die noch nicht beantworteten Interest-Pakete verwaltet. In CCNs werden nur die Interests geroutet. Datenpakete, die als Antwort darauf verschickt werden, verfolgen den Weg abschnittsweise entlang den PIT-Einträgen zurück. Wann immer ein Datenpaket auf einem Zwischenknoten zur Quelle eines entsprechenden Interest weitergeleitet wird, wird dessen Eintrag in der PIT gelöscht, da die Nachfrage beanwortet wurde. Erfolgt auf einen Interest keine Antwort, so kann der Eintrag in der PIT auch durch einen Timeout entfernt werden.

Geht ein Interest über eine Netzwerkschnittstelle ein, so wird zunächst eine Suche nach der längsten Übereinstimmung des Namens mit Einträgen in drei Speicherstrukturen durchgeführt. Findet sich ein identischer Eintrag im Pufferspeicher, so wird das Antwortpaket verschickt. Passiert das nicht, so wird zunächst verglichen, ob in der PIT bereits ein gleichlautender Name enthalten ist. Ansonsten wird das Paket über die FIB weitergeleitet, falls dort Hinweise zu finden sind, welche anderen Quellen befragt werden könnten und das Interest in der PIT abgespeichert. Enthält auch die FIB keinen übereinstimmenden Präfix, so kann die Anfrage als letzter Ausweg noch an die lokale Broadcastdomain weitergeleitet werden, bevor die Suche aufgegeben wird.

Wenn sich in der PIT bereits ein identisches Interest befindet, so wird lediglich der Sender der neuen Anfrage als zusätzliches Ziel hinzugefügt und das neue Interest verworfen.

Datenpakete können sehr einfach behandelt werden. Existiert ein entsprechender Eintrag in der PIT, so wird das Paket an die vermerkten Ziele weitergeleitet und der PIT-Eintrag gelöscht. Gibt es ein Paket dieses Namens im Pufferspeicher, so handelt es sich um ein Duplikat, das verworfen werden kann. Findet sich lediglich ein Ergebnis in der FIB, so ist das Paket nicht angefordert worden und kann ebenfalls vernichtet werden.

3. DATENVERKEHR

CCN ist darauf ausgelegt, unregelmäßige und unzuverlässige Verbindungen zu nutzen. Interest- und Datenpakete können daher auf dem Weg zu ihrem Ziel verloren gehen. Wird ein Interest nicht innerhalb eines gewissen Zeitrahmens befriedigt, so muss sich der Anfragende Teilnehmer selbst darum kümmern, das Paket neu anzufordern.

3.1 Datentransport

Da in einem CCN Anfragen über verschiedene Übertragungswege verschickt werden können, kann es vorkommen, dass ein Interest einen Knoten auf verschiedenen Wegen erreicht oder im Kreis läuft. Um zu verhindern, dass ein Antwortpaket auf beiden Wegen zurückgeschickt wird, enthalten Interestpakete einen zufällig generierten Erkennungswert. Kommt ein zweites Interest mit einem identischen solchen Wert an, so wird es verworfen.

Bei Datenpaketen ist dies nicht notwendig, da doppelte Pakete wie beschrieben verworfen werden. Somit können sie auch nicht im Kreis laufen.

3.1.1 Sicherstellen des Datenflusses

Auf ein Interest kommt im CCN höchstens ein Datenpaket. Ist dessen Größe nicht ausreichend, um die gesamten Daten zu transportieren, so muss ein neuer Interest gesendet werden, der den nächsten Teil in Relation zum ersten benennt. Dadurch gibt es ein Gleichgewicht zwischen Frage und Antwort. Wie beim TCP ist es allerdings möglich, mehrere Interests zu verschicken, bevor eine Antwort auf das erste kommt. Dadurch kann der Anfragende steuern, wie schnell er Daten empfangen kann. Geht ein Paket bei der Übertragung verloren, so kommt es nicht zu Stauungen, da alle weiteren Interest- und Datenpakete davon unabhängig sind.

Kommt es bei TCP zu Paketverlusten, so wird von den Endknoten die Größe des Übertragungsfensters dynamisch geändert, da davon ausgegangen wird, dass ein dazwischen liegender Router die Pakete aufgrund von Überlastung verworfen hat. Diese Problem wird von CCNs anders angegangen. Dort ist jeder einzelne Knoten im Netz selbst dafür verantwortlich, wie er mit hohen Lasten umgeht. Durch das LRU-Ersetzungsprinzip bleiben häufig benötigte Pakete im Netz, anstatt dass wie bei TCP alle Pakete eine FIFO-Queue durchlaufen müssen. So kommt es seltener zu Verspätungen beim Datenfluss.

3.1.2 Sequenzierung

Wie bei TCP ist es auch in einem CCN notwendig zu spezifizieren, welches Datenpaket der Empfänger als nächstes haben will. Dafür gibt es bei TCP das ACK-Feld im Paketheader: Sie geben die Sequenznummer des nächsten gewünschten Pakets an. Durch das Namensschema bei der Benennung von CCN-Paketen ist hier ein Ansatz, der auf einem reinen Zahlensystem beruht, nicht möglich.

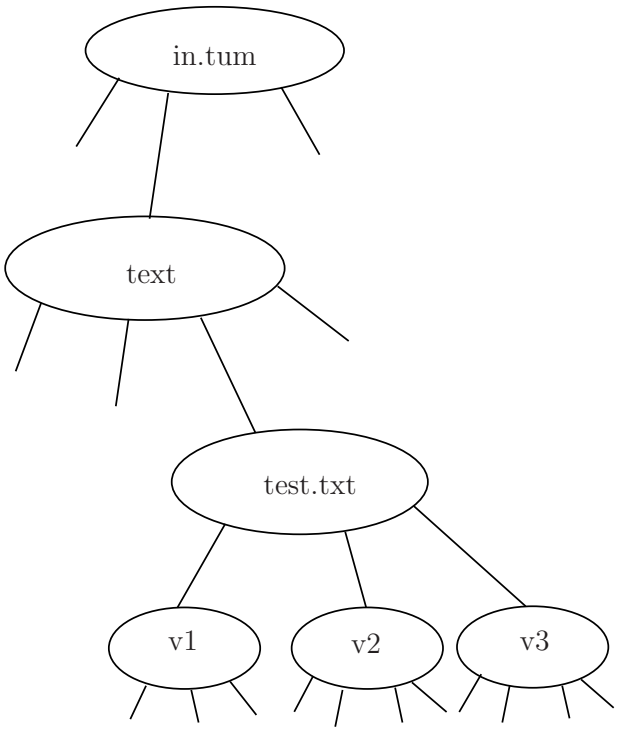


Abbildung 2: Hierarchisches Namensschema

Stattdessen wird hier die hierarchische Struktur beim Aufbau der Namen ausgenutzt. Auch wenn die Namen eine Bedeutung auf höherer Ebene haben, so ist es für den Transport der Daten irrelevant, wie dieser Name tatsächlich aussieht. Aus Sicherheitsgründen kann die Benennung sogar verschlüsselt sein.

Wenn der vollständige Name eines Pakets nicht bekannt ist, so kann der Name in Relation zu einem Vorgängerpaket angegeben werden. Durch die Traversierung des Namens anhand einer Baumstruktur ist es etwa möglich, den ersten Kindknoten einer Ebene anzufordern, oder dessen rechten Nachbar. Ein Beispiel (siehe Abbildung 2) wäre es, von einer Datei `in.tum/text/test.txt` die aktuellste (von einer unbekannten Anzahl) Version anzufordern, indem man ein Interest mit dem Namen `in.tum/text/test.txt/rechtesKind` versendet. Darauf würde als Antwort das erste Paket der aktuellsten Version folgen.

Auch wenn das eigentliche Aufbauschema dieser Namen nicht direkt im Protokoll spezifiziert wird, sollen sich hier noch Standards entwickeln, die zu einem leicht umsetzbaren und transparenten System führen, um Inhalte anzusprechen.

3.1.3 Nutzung verschiedener Netzwerkschnittstellen

CCN-Pakete können sich, wie bereits beschrieben, nicht im Kreis bewegen. Daher können, im Gegensatz zu IP, mehrere Netzwerkschnittstellen gleichzeitig (beispielsweise UMTS

neben einer WLAN-Verbindung) ohne zusätzlichen Aufwand benutzt werden. Dies soll eine Reihe von Vorteilen mit sich bringen: So soll neben einer höheren Geschwindigkeit auch eine höhere Stabilität der Verbindung und erweiterte Mobilität erreicht werden. Erreicht man beispielsweise mit einem mobilen Gerät die Grenzen des vom bisher verwendeten WLAN abgedeckten Bereichs, so etwa kann eine Datenübertragung etwa über UMTS nahtlos über die verbleibende Verbindung fortgesetzt werden.

Darüber hinaus sollen alle an einer Übertragung beteiligten Knoten durch das Erhalten von Paketen mit gleichen Namenspräfixen über verschiedene Schnittstellen Informationen erhalten, was die „beste“ Schnittstelle ist, um Interests mit diesem Präfix weiterzuleiten. Diese Präferenzen für bestimmte Schnittstellen werden in der FIB für jeden Präfix einzeln gespeichert. Je nach gewählter Strategie können einzelne Interests dann wahlweise nur über den „besten“ Weg oder über mehrere weitergeleitet werden. Zudem werden für die einzelnen Anschlüsse auch Attribute vorgehalten wie *isContentRouter* oder *BroadcastCapable*.

Zusammen werden diese Informationen als *Strategy Layer* des CCN-Protokolls bezeichnet. Die Testimplementierung von Jacobson [3] wählt in der Standardeinstellung die Strategie, Interests zuerst an alle Broadcast-fähigen Schnittstellen zu senden und dann, falls nötig, alle weiteren sequentiell durchzuprobieren. So soll erreicht werden, dass Daten zuerst im lokalen Netz gesucht werden und nur die Interests, die so nicht erfüllt werden können, über Router weitergeleitet werden.

Schnittstelleneinträge in einer FIB kommen auf verschiedene Weisen zustande. Wird eine Datenquelle mit einem CCN verbunden, so führt sie eine *register*-Operation aus, um den verbundenen Knoten mitzuteilen, zu welchen Präfixen Inhalte bereitgehalten werden. Die verbundenen Knoten speichern dies dann in ihrer FIB ab. Zusätzlich wird in einem Flag gespeichert, ob diese Informationen auch weiterverbreitet werden sollen. Dies kann durch sogenannte *announcement agents* geschehen, die die lokale FIB nach solchen Präfixen durchsuchen. Übertragen werden können diese Informationen wieder über CCN oder etwa auch über IP.

3.2 Routing

Das Routing von CCNs soll problemlos über die vom IP bekannten Strukturen möglich sein. Jacobson sieht sein Protokoll als idealen Weg, geroutete Informationen zu vermitteln. Auch das soll ein Hilfsmittel sein, CCN-Infrastruktur neben herkömmlichen IP-Netzen aufzubauen und diese schrittweise abzulösen.

3.2.1 Lokales Routing

Beim Intra-Domain Routing sind etwa das OSPF- [1] und das IS-IS-Protokoll [2] verbreitet. Bei IS-IS werden adjazente Knoten in Abhängigkeit ihrer MAC-Adresse (OSI-Layer 2 [4]) verwaltet, aber Inhalte über IP-Präfixe auf OSI-Layer 3 angesprochen. CCN soll sich sehr ähnlich wie IP darauf verhalten. Auch wenn die Struktur der angesprochenen Namenspräfixe bei CCN sich deutlich von ihren Pendants bei IP unterscheiden, soll durch Ausnutzung eines *type label value (TLV)*-Feldes sichergestellt werden, dass CCN-fähige Router problemlos mit bestehenden Netzen zusammenarbeiten. Die Spezifikation sieht vor, dass unbekannte TLV-Wert ignoriert werden, wodurch die Kompatibilität gewährleistet ist.

Durch das Konzept werden in einem CCN prinzipbedingt sehr gute Werte für benötigte Bandbreiten und Latenzzeiten erreicht, da seltener gleiche Daten die gleichen Strecken zurücklegen müssen.

Ein unterschiedliches Verhalten von IP und CCN beim Routing gibt es, wenn ein Router mehrere unterschiedliche Teilnehmer kennt, die einen bestimmten Namenspräfix auflösen können. Da bei IP Daten im Kreis laufen können und daher Routing in Form von Spannbäumen erfolgen muss, kann nur ein einziger Knoten angesprochen werden, um Pakete weiterzuleiten. Dagegen leitet ein CCN das Interest an alle möglichen Stellen weiter. Dies liegt daran, dass in einem CCN nicht der gesamte Datenbestand mit einem bestimmten Präfix auch über eine einheitliche Adresse erreichbar ist. Ermöglicht wird das durch die Eigenschaft, dass sich Pakete nicht im Kreis bewegen können.

Auch wenn CCNs keine Spannbäume für Routingeinträge berechnen müssen, sind die Routinginformationen vollständig. Um die Funktionalität herzustellen, sollen lediglich Änderungen der jeweiligen Implementierungen der Routingprotokolle nötig sein, nicht jedoch Änderungen an den Protokollen selbst.[2]

3.2.2 Netzübergreifendes Routing

Kämem bei einer gewissen Anzahl von Kunden eines Internetproviders CCNs zum Einsatz, läge es für den ISP nahe, dies auch zu unterstützen. Dadurch könnte der durchgehende Datenverkehr bei häufig abgerufenen Daten deutlich reduziert werden.

Allerdings existieren bisher noch keine konkreten Wege, wie der Datenverkehr in diesen Dimensionen geroutet werden könnte. Insbesondere was den Datenaustausch zwischen zwei CCN-fähigen Domains betrifft, wenn zwischen diesen ein Router, der dazu nicht in der Lage ist betrieben wird, ist noch ein Problem. Zwar kann man CCN-Pakete etwa auch über einen UDP-Tunnel versenden, aber das wirft Probleme auf und widerspricht auch dem Grundprinzip des CCN.

4. SICHERHEIT

Das Sicherheitskonzept von CCNs richtet wie das Netzwerk selbst den Fokus auf die Daten. Daher werden nicht (wie etwa beim Einsatz von https in IP-Netzen) die Verbindungen zwischen zwei Teilnehmern gesichert (was im Widerspruch dazu stehen würde, dass Daten über mehrere Wege übertragen werden und sogar von mehreren Quellen kommen können), sondern die Daten selbst. Da Daten nicht nur auf bestimmten Servern bereit gestellt sind, sondern auch Duplikate an anderen Positionen im Netz existieren, besteht eine erhöhte Gefahr, dass diese manipuliert werden. Daher werden grundsätzlich alle Daten mit Zertifikaten validiert, um sicherzustellen, dass sie auch aus der richtigen Quelle stammen. Darüber hinaus können die Daten und sogar, wie

[2]Laut Jacobson gilt dies allerdings nur für Link-State-Protokolle wie IS-IS oder OSPF, nicht jedoch für Protokolle, die auf Distanzvektoren beruhen wie RIP.

bereits beschrieben, die Namen selbst auch noch verschlüsselt werden.

4.1 Validierung

Bei der Signierung der Datenpakete werden neben den Daten auch die Namen selbst mit einbezogen. So soll sichergestellt werden, dass die Inhalte und deren Namen zusammenpassen. Außerdem wird es dadurch unnötig, die Namen zusätzlich mit der Signatur der Daten zu verschlüsseln, wie es bei anderen Validierungsmethoden erforderlich ist.

Die Signaturen selbst basieren auf den gewöhnlichen Public-Key-Strukturen, so dass jeder Knoten im Netzwerk die erhaltenen Daten auch überprüfen kann. Die Signaturart kann der Veröffentlichende je nach Einsatzzweck und benötigter Sicherheit selbst festlegen. Jedes CCN-Datenpaket enthält Informationen, wie der zum Validieren nötige Schlüssel zu erhalten ist. Allerdings ist keineswegs gewährleistet, dass Daten mit einem bisher unbekannten Schlüssel auch tatsächlich von der Person veröffentlicht wurden, die als Quelle benannt wird. Die Signatur stellt lediglich sicher, dass alle Pakete ursprünglich von derselben Person stammen und lässt so einen Schluss auf die Konsistenz zu.

4.2 Trust Management

Das Konzept der mit den Inhalten in die Signatur einfließenden Namen bringt einen sehr nützlichen Nebeneffekt mit sich: Wenn ein Public Key selbst auf diese Weise in ein Datenpaket gesteckt wird, entsteht dadurch ein Zertifikat. So kann auf einfache Art eine Infrastruktur geschaffen werden, bei der Inhalte andere Inhalte validieren.

Neben dieser Struktur eines Netzwerks von vertrauenswürdigen Quellen können auch weiter herkömmliche Modelle wie die bekannte PKI verwendet werden oder neue, an CCNs angepasste. Ein gut zur CCN-Struktur passendes Modell ist das von SDSI/SPKI. Hierbei wird es ermöglicht, dass Schlüssel aus einem gemeinsamen *namespace* sich gegenseitig authentifizieren können. Beispielsweise könnte ein Unternehmen seine Angestellten validieren oder umgekehrt, von einem vertrauenswürdigen Mitarbeiterschlüssel auf seine Kollegen und das Unternehmen geschlossen werden.

Ein weiterer Sicherheitsmechanismus ist das Verfolgen von Vertrauenswegen. So kann man von bereits bekannten vertrauenswürdigen Quellen, die eine andere Quelle identifizieren, auf deren Vertrauenswürdigkeit schließen.

Hat ein Client noch keine Informationen über den Public Key zu angeforderten Daten, so lässt sich dieser immer anhand des Datenpakets finden. Dort befinden sich im Header Felder, anhand derer der Urheber zu indentifizieren ist und wo sich dessen Public Key finden lässt.

4.3 Verschlüsselung

Da Verschlüsselung das einzige Mittel ist, um in einem CCN eine Zugangskontrolle durchzuführen, ist das Protokoll in Bezug auf Verschlüsselung von Daten sehr flexibel. Es können alle gängigen Verfahren angewendet werden. Da auch die Namen der Pakete mitverschlüsselt werden können, ist es auch möglich, Schlüssel zur Dekodierung über ein CCN zu verschicken.

4.4 Netzwerksicherheit

Viele heute gängige Angriffsmethoden auf Netzwerke lassen sich in einem CCN nicht oder nur schwer durchführen. Da die Kommunikation darauf basiert, *über Daten* zu sprechen, kann auf diese Weise nicht direkt *zu einem bestimmten Host* gesprochen werden. Daher können nur schwer bösartige Pakete gezielt an bestimmte Teilnehmer verschickt werden. Ein effizienter Angriff sollte daher eher ein Denial of Service-Schema verfolgen, indem Inhalte „versteckt“ werden (d.h. nicht weitergereicht) oder das Verhindern der Zustellung von Paketen durch Überfluten von Knoten mit nutzlosen Paketen.

Durch die Signierung von Paketen und eine überprüfbare Kette von Public Keys sind Nutzer grundsätzlich in einem gewissen Umfang vor Angriffen durch manipulierte Daten geschützt.

Blinde DoS-Attacken durch Bombardieren eines Knotens mit Datenpaketen werden in CCNs durch das Gleichgewicht zwischen Interest- und Datenpaketen verhindert. Der Datenverkehr verteilt sich dabei gleichmäßig auf mehrere Teilnehmer, die Daten aber nur nach einem vorhandenen Interest weiterleiten. Ist dieses nicht vorhanden, so wird das Pakete auch nicht weitergeleitet.

Eine mögliche Angriffsart wäre das Überfluten eines Ziels mit Interests. Um das zu erreichen, müsste allerdings eine sehr große Zahl an Interests generiert werden, die alle einen individuellen Namen besitzen. Ansonsten kämen Duplikate nie beim Ziel an und würden nur eine zusätzliche Last für den Angreifer bewirken. Diesen Angriffen stehen zwei Dinge entgegen: Zum einen können Router selbstständig die Menge an Interests reduzieren, die an eine Domain weitergeleitet werden, wenn auf diese keine Antwort folgt[3]. Zum anderen kann ein Teilnehmer auch selbst verbundene Router bitten, die Menge an Anfragen mit einem bestimmten Namenspräfix zu reduzieren.

Nicht zuletzt existiert auch noch die Möglichkeit, den Lauf von Daten zu beeinflussen. So kann verlangt werden, dass Daten nur über Router verschickt werden, die deren Integrität mittels Public Key des Erstellers überprüfen.

5. PRAKTISCHE UMSETZUNG & KRITIK

Van Jacobson und sein Team am PARC haben eine Testimplementierung des CCN-Protokolls erstellt [3]. Diese ist lauffähig, hat aber noch diverse Einschränkungen.

5.1 Ergebnisse

Bei Geschwindigkeitsmessungen schneidet CCN noch etwas langsamer ab als eine TCP-Verbindung mit den selben übertragenen Daten. Dies ist zum Teil auch der Tatsache geschuldet, dass Daten für Tests in UDP-Pakete verpackt werden müssen, um über bestehende Strukturen geroutet werden zu können [6].

Wenn mehrfach gleiche Daten über ein Netzwerk verschickt werden, steigt der Aufwand in einem IP-Netz linear mit der Anzahl der Abfragen. Bei der Testimplementierung eines

[3]Bei einer solchen Zahl an zufällig generierten Interests gibt es sehr wahrscheinlich nur wenige Antworten

CCN hat sich wie erwartet gezeigt, dass für den ursprünglichen Sender der Aufwand dagegen konstant der gleiche ist, wie bei einer einzigen Anfrage.

Neben weiteren Versuchen beinhalten diese Tests auch eine Implementierung eines Voice-over-CCN-Protokolls, das im Rahmen der Tests gut funktioniert und es ermöglicht, trotz wechselnder Verbindungen ein Telefongespräch ohne Paketverluste aufrecht zu erhalten [5].

5.2 Kritik

Trotz der erfolgreichen Testimplementierung gibt es aber auch Kritik, sowohl am Konzept der CCNs, als auch an deren Umsetzung. Wie bereits in Abschnitt 3.2.2 angesprochen, gibt es noch keine Implementierung für das Routing auf Providerebene. Jacobson lässt in seiner Ausarbeitung des Konzepts offen, wie es funktionieren soll und beschreibt lediglich vage Möglichkeiten.

Das Konzept eines Content Centric Network sieht vor, dass lediglich Daten direkt angesprochen werden, nicht deren Speicherort. Allerdings widerspricht sich die Umsetzung dabei selbst. Denn um bestimmte Daten zuverlässig zu erhalten, genügt es nicht, einen Knoten im Netzwerk anzufragen, der Informationen zu Daten mit einem bestimmten Präfix hat. Theoretisch kann jeder einzelne Rechner im Netz die gewünschten Informationen bereithalten. Da allerdings nicht jeder per Broadcast gefragt werden kann, kann der Suchende die Daten möglicherweise nie erhalten, weil er keine Informationen darüber hat, über welche Schnittstelle er danach fragen kann und auch keinen anderen Knoten kennt, der mehr darüber weiß. Daher spricht Jacobson immer wieder von einer Aufteilung in Domains, die aber der Idee widerspricht, dass Daten unabhängig von bestimmten Orten auffindbar sein sollen.

Ein weiteres Problem, das im Konzept nicht ausreichend betrachtet wird, ist die Konsistenz der Daten in großen Netzwerken. So ist es durchaus möglich, dass unterschiedliche Inhalte mit identischen Namen auftauchen. Zum Teil kann dieses Problem zwar durch die Signierung der Daten behoben werden (so wird sichergestellt, dass nur berechtigte Personen gültige Daten zu bestimmten Präfixen anbieten können), allerdings ist nicht sichergestellt, dass geänderte Daten an alle Knoten durchdringen, die Informationen zum entsprechenden Präfix liefern können. So kann beispielsweise nicht gewährleistet werden, dass die Daten, die über eine bestimmte Verbindung empfangen wurden, auch tatsächlich der aktuellsten verfügbaren Version entsprechen. Und wenn etwa über verschiedene Schnittstellen gleichzeitig mehrere Interests zu Teilen der aktuellsten Version einer Datei angefordert werden, dann kann es passieren, dass durch unterschiedlichen Versionsstand eines der Kontakte die erhaltenen Daten letztlich völlig unbrauchbar sind. Bestimmte dynamisch erzeugte Daten sollten auch niemals aus dem Cache eines bestimmten Netzwerkknotens

Die größte Schwäche am Konzept hat aber mit der praktischen Realisierbarkeit im großen Rahmen zu tun. Gelingt es noch, in kleinem Maßstab ein funktionierendes Netz aufzubauen mit einigen wenigen verbundenen Rechnern und einem überschaubaren Maß an Daten, so wirft dies auf Internetebene große Probleme auf: Die beteiligten Knoten müssen große Mengen an Speicherplatz darauf verwenden, um die benötigten Datenpuffer zu realisieren. Zudem wird gerade an den großen Internetknotenpunkten wie etwa den Enden von Interkontinentalkabeln die Anzahl der benötigten Einträge in der FIB und der PIT unüberschaubar groß.

6. FAZIT

Das Konzept eines inhaltszentrierten Netzwerks ist an sich eine gute Idee. Immerhin ist es für Anwender in aller Regel irrelevant, woher die angeforderten Daten kommen, solange sie geliefert werden. Durch ein CCN kann in der Theorie sowohl die Geschwindigkeit als auch die Sicherheit im Netz deutlich steigen. Allerdings wirft die Implementierung außerhalb der überschaubaren, lokalen Broadcastbereiche unüberwindbare Hindernisse auf. Gerade die großen Knotenpunkte des Internet könnten den notwendigen Aufwand nicht bewältigen, was das Konzept zum Scheitern verurteilt, bevor es wirklich eingesetzt wurde.

7. LITERATUR

[1] IETF. RFC 2328 - OSPF Version 2.

[2] IEFT. RFC 3787 - Recommendations for Interoperable IP Networks using Intermediate System to Intermediate System (IS-IS).

[3] Project CCNx. http://www.ccnx.org, 2010.

[4] ISO/IEC 7498-1. *Open Systems Interconnection: Basic Reference Model: The Basic Model.*

[5] V. Jacobson, D. K. Smetters, N. H. Briggs, M. F. Plass, J. D. Thornton, and R. L. Braynard. Voccn: Voice-over content-centric networks. *ReArch'09*, 2009.

[6] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking named content. *CoNEXT'09.*

OpenFlow

Sebastian Rampfl
Betreuer: Dipl.-Ing. Dirk Haage
Seminar Innovative Internet Technologien und Mobilkommunikation SS2010
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: rampfl@in.tum.de

ABSTRACT

Das Internet befindet sich im ständigen Wandel. Neue Angebote entstehen, andere verschwinden wieder. Auch neue Technologien und Netzwerk-Protokolle werden entwickelt. Doch diese zu etablieren und in das Internet zu integrieren erweist sich als schwierig, nicht zuletzt deswegen, weil es den Forschern an Möglichkeiten fehlt, ihre Ideen und Technologien unter realtitätsnahen Bedingungen zu testen. OpenFlow ist ein neues Protokoll, welches genau das bietet. Es gibt Forschern Kontrolle über bestehende Netz-Infrastruktur und ermöglicht ihnen somit ihre Ideen in größeren Netzen zu testen, und das ohne, dass der normale Traffic der Infrastruktur, also der Datenverkehr, der im normalen Betrieb entsteht, davon beeinflusst wird. Doch nicht nur für die Forschung ist OpenFlow ein vielversprechendes Protokoll, denn diese neue Möglichkeit der Einflussnahme auf ein Netzwerk eröffnet eine ganze Reihe weiterer Möglichkeiten für technische Innovationen.

Keywords

OpenFlow, Virtualisierung, FlowVisor

1. EINLEITUNG

Technischer Fortschritt und Innovationen auf dem Gebiert der Netzwerktechnik und der Netzdienste sind wichtige Faktoren für den Ausbau und die Optimierung von Netzwerken und nicht zuletzt des Internets. Wer sich mit Netzen und den verwendeten Techniken und Protokollen auseinandersetzt stellt fest, dass sich neue experimentelle Protokolle meist nur schwer in bestehende Netzinfrastruktur integrieren lassen. Man spricht hier von der „Verknöcherung“ von Netzwerken oder allgemein des Internets. Das hat zum einen den Grund, dass sich Protokolle nicht einfach durch neue ersetzen lassen. So kann beispielsweise in einem Netzwerk nicht auf einem Teil der Infrastruktur das IP-Protokoll laufen während auf anderen Geräten innerhalb des Netzes ein alternatives IP-Protokoll zum Einsatz gebracht wird. Die in Netzwerken eingesetzten Geräte werden von den Herstellern zudem größtenteils als geschlossene, nicht programmierbare Systeme konzipiert, die kaum Einflussnahme auf die darauf laufenden Protokolle ermöglichen. Gründe dafür sind zum einem der Wunsch der Hersteller die Architektur und Funktionsweise ihrer Hardware zu verschleiern um ihren Wettbewerbern nicht unnötig Informationen über ihre Technologien zu offenbaren, zum anderen besteht die berechtigte Befürchtung das Veränderungen an ihren Geräten durch Forscher die Stabilität ganzer Netzwerke gefährdet seien könnte. Alle diese Umstände führen zu der Erkenntnis, dass das Implementieren und Testen experimenteller Protokolle in bestehender Netz-Infrastruktur wie zum Beispiel einem Campus-Netzwerk kaum möglich ist. Alternativ gäbe es auch die Möglichkeit ein Netz aus programmierbaren Switches und Routern aufzubauen, um auf dieser Hardware die neuen Protokolle laufen zu lassen. Dies hätte den Vorteil, dass diese Variante, abgeschottet von anderen Netzen, wie einem Campus-Netzwerk, normalen Traffic nicht negativ beeinflussen kann. Ein solches reines Forschungs-Netz kommt für die meisten Forscher und Forschungseinrichtungen allerdings nicht in Frage, da es mit zu hohen Kosten verbunden ist wenn das Netzwerk eine Größe und Portdichte erreichen soll, die ein Testen unter realitätsnahen Bedingungen ermöglicht. Genau hier setzt OpenFlow an.

In diesem Paper werden das OpenFlow-Protokoll, seine Funktionsweise und die Möglichkeiten, die es für die Forschung und weitere Anwendungen bietet, beschrieben. Das Paper ist dabei wie folgt gegliedert. In Kapitel 2 werden zuerst die Anforderungen erläutert, die sich aus den in Kapitel 1 geschilderten Problemen ableiten. Im Anschluss wird auf die technische Umsetzung von OpenFlow und die Komponenten dieses Systems eingegangen. Dabei wird auch erklärt wie und in welchem Maß die Implementierung von OpenFlow die gestellten Anforderungen erfüllt. Zusätzlich wird noch eine Erweiterung zu OpenFlow erläutert, der so genannte FlowVisor-Controller. In Kapitel 3 beschreibt das Paper dann OpenFlow in der Praxis. Dazu werden zuerst verschiedene Forschungsansätze diskutiert und im Anschluss weitere Beispiele für Arten der Anwendung genannt. Zusätzlich werden in Kapitel 3 noch drei in ihrer Entwicklung weit fortgeschrittene und auf OpenFlow basierende Projekte beschrieben, um noch einmal die umfangreichen Möglichkeiten, die OpenFlow bietet, zu verdeutlichen. In Kapitel 4 werden dann noch allgemeine Fakten zum OpenFlow-Projekt genannt und in Kapitel 5 ein Fazit gezogen.

2. Anforderungen und technische Umsetzung

2.1 Anforderungen an OpenFlow

Die erste Anforderung an das OpenFlow-Protokoll war es, ein Einbinden von Standard-Netz-Hardware zu ermöglichen. Es sollten für den Betrieb also keine speziellen Geräte wie teure offene Systeme notwendig sein. Dadurch wird sichergestellt, dass OpenFlow in normaler Netzwerk-Infrastruktur wie einem Universitäts-Netzes zum Einsatz kommen kann. Gleichzeitig sollte es aber den in einem solchen Netzwerk anfallenden Traffic nicht beeinflussen, d.h. Tests von experimentellen Protokollen dürfen die Stabilität und Zuverlässigkeit des Netzes nicht gefährden. Zusätzlich sollten in einem OpenFlow-Netz Tests zu einer möglichst breiten Palette an Forschungsansätzen realisierbar

sein. Auch auf die Hersteller der Hardware musste dabei Rücksicht genommen werden. Sie dürfen dabei ihre internen Switch- und Router-Implementierungen nicht offenlegen müssen. Im Folgenden wird darauf eingegangen wie diese Anforderungen umgesetzt wurden. Die in Kapitel 1 beschriebenen Probleme und die daraus resultierenden Anforderungen werden in Referenz [1] erläutert.

2.2 Das OpenFlow-Konzept und die technische Umsetzung

Die Idee hinter OpenFlow ist folgende: Über einen Rechner im Netzwerk, dem sogenannten Controller, werden die FlowTables aller OpenFlow-fähigen Switches, Router und Access Points in einem Netzwerk kontrolliert. Das OpenFlow-Protokoll schafft für den Controller eine Schnittstelle in allen Geräten der Netz-Infrastruktur um Flows, also Paketströme, durch das Netzwerk zu lenken. Wie in Abbildung 1 zu sehen, erfolgt das Editieren der FlowTable über die eigens dafür geschaffene Schnittstelle SecureChannel. Diese Schnittstelle muss allerdings schon vom Hersteller der Netzwerk-Geräte ab Werk integriert sein. Damit ist die Anforderung nach der Einsetzbarkeit in einer normalen Netzwerk-Infrastruktur nur bedingt erfüllt, da die Geräte erst ausgetauscht werden müssen, um OpenFlow innerhalb eines bestehenden Netzwerks einsetzen zu können. In einigen Fällen ist es auch möglich durch ein Firmware-Update, also das Aktualisieren der auf der Hardware installierten Betriebssysteme, das OpenFlow-Protokoll in ein Gerät zu integrieren. Im Folgenden wird auf die einzelnen Bestandteile und die genauere Funktionsweise von OpenFlow, wie sie in dem offiziellen Whitepaper [1] und in der OpenFlow-Spezifikation [2] beschrieben ist, eingegangen.

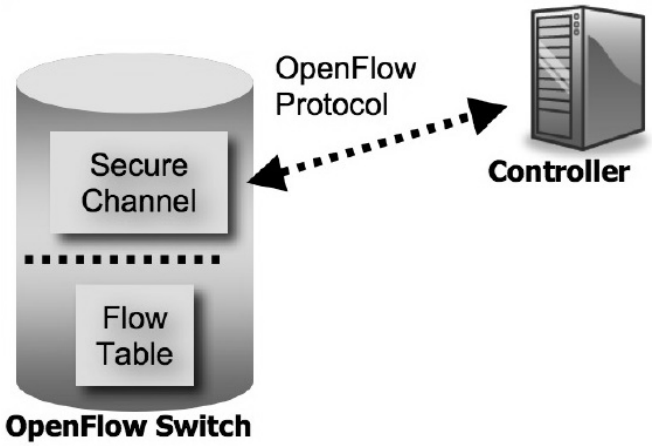


Abbildung 1. FlowTable, SecureChannel, Controller [2]

2.2.1 Flows und FlowTable

Ein Flow ist ein Paketstrom. Dabei wird der Flow durch die Daten im Header der Pakete definiert. Beispielsweise können alle Pakete mit derselben IP-Zieladresse einem Flow zugeordnet werden. Man kann einen Flow aber auch anhand von Ports oder MAC-Adressen festlegen. Grundsätzlich kann jede Information im Header als Regel zur Definition eines Flows dienen. Die FlowTable listet die Flows auf und legt für jeden Flow eine oder mehrere Regeln fest wie das Switch oder der Router mit dem Flow verfahren soll. Wie in Abbildung 2 zu sehen besteht eine FlowTable aus drei Spalten. Header Fields gibt an welche Pakete zu diesem Flow gehören. Im Feld Counters werden Statistiken über den Flow geführt und im Feld Actions finden sich die Befehle die auf die Pakete dieses Flow angewendet werden sollen. So kann das Paket beispielsweise an einen bestimmten Port dirigiert werden, es kann aber auch verworfen werden. Die OpenFlow-Spezifikation legt hier eine Reihe von Befehlen fest die in jedem OpenFlow-fähigen Gerät implementiert seien müssen.

Abbildung 2. Die Spalten der FlowTable

2.2.2 OpenFlow-Switch

Die Spezifikation unterscheidet hier zwei unterschiedliche Typen von Switches. Das erste ist das dedicated OpenFlow-Switch, auch OF-only Switch genannt. Es kennt nur die drei wichtigsten Befehle. Der erste Befehl leitet ein Paket zu einem Switch-Port, d.h. zu einer Datenleitung die aus dem Switch zu einem anderen Gerät führt, weiter. Der zweite Befehl sendet das Paket zum Controller und der Dritte verwirft das Paket. Die zweite Variante ist das OF-enabled Switch. Es verfügt über denselben Befehlssatz und zusätzlich noch über die Möglichkeit Pakete an die normale „Processing pipeline“ des Gerätes weiterzuleiten, d.h. die Routing-Entscheidung wird vom Switch selbst ohne Einfluss des OpenFlow-Controllers getroffen. Diese Funktion ist notwendig um normalen Traffic von Forschungs-Traffic zu trennen. Letzterer wird vom Forscher über die FlowTable kontrolliert während der normale Traffic unbeeinflusst davon durch die Processing Pipeline des Switches oder Router bearbeitet und weitergeleitet wird. Der Netzwerk-Administrator legt hierbei durch VLANs fest welche Pakete welchen Traffic zuzuordnen sind und der Forscher hat somit keinen Einfluss auf den normalen Traffic, da dieser nicht in der FlowTable erfasst werden kann. Damit wird schon eine zentrale Anforderung an OpenFlow, die strikte Trennung von Forschungs-Traffic und dem durch den normalen Gebrauch der Netz-Infrastruktur entstehenden Traffic, erfüllt. Die Spezifikation bietet noch eine Reihe weiterer optionaler Befehle an, die nach Ermessen der Hersteller implementiert werden können. So lässt sich beispielsweise auch die VLAN-ID eines Flows modifizieren.

2.2.3 SecureChannel und Controller

Der SecureChannel ist die Schnittstelle zwischen Switch und Controller. Diese Schnittstelle muss im Switch implementiert sein und kann immer nur mit einem Controller kommunizieren. Allgemein kann es in einem OpenFlow-Netzwerk nur einen Controller geben. Der Controller erstellt, editiert und löscht FlowTable-Einträge, d.h. er definiert Flows und legt fest wie mit ihnen verfahren werden soll. Der Forscher implementiert sein experimentelles Protokoll auf dem Controller und dieser steuert den Forschungs-Traffic über die Switches und Router im Netzwerk dann nach den Vorgaben des experimentellen Protokolls. Im Detail passiert dabei Folgendes. Der Forscher sendet von einem Host im Netzwerk Pakete. Das erste OF-Switch, das diese Pakete erhält leitet sie weiter zum Controller, da es die Pakete noch keinen Flow zuordnen kann. Der Controller analysiert den Header der Pakete, definiert diese dann als Flow und schafft entsprechende Einträge in den FlowTables.

Gleichzeitig fügt er diesen Einträgen noch die Befehle hinzu, welche auf den Flow angewandt werden sollen. Nun bewegen sich die Pakete des Forschers nach den von seinem experimentellen Protokoll definierten Regeln durch das Netzwerk. Der Controller kann die von der FlowTable geführten Statistiken zu den Flows abrufen und so die Leistung des getesteten Protokolls analysieren. Hier wird schon ein wichtiger Vorteil von OpenFlow im Vergleich zu einem wie in Kapitel 1 beschrieben programmierbarem Netzwerk deutlich. In Letzterem muss das experimentelle Protokoll auf jeden Switch und Router implementiert werden. Die Implementierung muss dabei den Hersteller-spezifischen Eigenheiten der einzelnen Geräte angepasst werden. In einem OpenFlow-Netzwerk wird das experimentelle Protokoll nur auf dem Controller implementiert, welcher dann durch die standardisierte Schnittstelle in den Geräten die entsprechenden Änderungen durchführt. Die Hersteller müssen dazu die internen Implementierungen ihrer Software und den genauen Aufbau ihrer Hardware nicht offenlegen. Somit wird eine weitere Anforderung an OpenFlow erfüllt. Für den Einsatz von OpenFlow in beispielweise einen Campus-Netzwerk ist es nicht notwendig, dass in jedem Switch, Router oder Access Point das OpenFlow-Protokoll integriert ist. Der Controller hat dann aber nur Kontrolle über die OpenFlow-Geräte und der Forschungs-Traffic kann nur auf diesen Geräten nach den vom dem Controller definierten Regeln beeinflusst werden. Umso größer der Anteil an OpenFlow-Geräten in einem Netzwerk umso komplexere und realitätsnähere Versuche lassen sich damit umsetzen. Die sichere Kommunikation zwischen dem Controller und den SecureChannels der Geräte wird durch den Austausch von Zertifikaten zur Authentifizierung und dem Einsatz von TLS zu Verschlüsselung gewährleistet. Dies ist notwendig um zu verhindern, dass sich ein Teilnehmer im Netzwerk unrechtmäßig Kontrolle über die Switches und Router verschafft. Neben der Sicherheit stellt sich auch die Frage ob ein einzelner Controller überhaupt alle Flows in akzeptabler Geschwindigkeit bearbeiten kann. Die OpenFlow-Autoren kommen aufgrund von Tests zu dem Schluss, dass dies der Fall ist, denn ein gängiger PC kann schon mit bis zu 10.000 Flows in der Sekunde umgehen. Der zentralisierte Controller schafft eine einheitliche Sicht auf das Netzwerk, seine Topologie, seine Auslastung und Veränderungen im Netz und eröffnet dadurch neue Möglichkeiten die vorher nur schwer oder überhaupt nicht zu realisieren waren.

2.2.4 FlowVisor

OpenFlow ist so konzipiert, dass nur ein Controller das OpenFlow-Netz steuert, d.h. es kann immer nur ein experimentelles Protokoll getestet werden. Diesem Problem widmet sich Referenz [3] und liefert als Lösung den FlowVisor-Controller. FlowVisor ist eine Software für den Controller, die es Forschern ermöglicht gleichzeitig mehrere Controller zu betreiben um unterschiedliche Protokolle gleichzeitig im Netzwerk zu testen. Wie in Abbildung 3 zu sehen fungiert FlowVisor hier als transparenter Proxy für weitere Controller. Dazu werden so genannte Slices eingerichtet. Sie werden ähnlich wie Flows definiert, d.h. Pakete werden einem Slice aufgrund ihrer Header-Informationen zugeordnet. Das können beispielsweise alle Pakete mit derselben MAC-Ziel-Adresse, desselben TCP-Quell-Ports oder einer Kombination daraus sein. Die Slices müssen klar voneinander abgetrennt sein und dürfen sich nicht überschneiden. Zusammenfassend kann man sagen, dass Slices aus einem oder mehreren Flows bestehen auf denen exklusiv ein Experiment durchgeführt wird. Wichtig bleibt dabei zu erwähnen, dass FlowVisor diese Slices definiert und dafür sorgt, dass das ein Paket nicht zwei unterschiedlichen Slices zugeordnet werden kann. Jedem mit dem FlowVisor-Controller verbundenen Controller werden ein oder mehrere Slices zugeteilt, die er dann für seine Experimente nutzen kann. Diese können wiederum auch FlowVisor-Controller sein, die die Slices nochmal in weitere Slices aufteilen und weiteren Controllern zuteilen. Dieses Konzept nennt sich Rekursive Delegation. FlowVisor überprüft jede über ihn verschickte OpenFlow-Steuerungs-Nachricht und kann damit auftretende Probleme erkennen und ggf. beheben. Probleme können beispielsweise entstehen wenn ein Controller Flows definiert, die nicht mehr in dem ihm zugeteilten Slice liegen, d.h. er definiert Regeln für Pakete, die ggf. zum Forschungs-Traffic eines anderen Controllers gehören und sich damit beide Controller gegenseitig ihre Protokoll-Tests verfälschen. Der FlowVisor-Controller kann dadurch auch gewisse Verhaltensregeln im Umgang mit dem Forschungs-Traffic und den FlowTables durchsetzen. Die Funktionalität die FlowVisor bietet hätte auch in OpenFlow integriert werden können. Die OpenFlow-Autoren haben das aber bewusst unterlassen, denn dies hätte genau wieder zu der in Kapitel 1 beschriebenen Verknöcherung des Netzes geführt, die die Entwicklung von OpenFlow erst notwendig gemacht hat. Sie streben eine Modularisierung ihres Systems an, in der einzelne Komponenten wie FlowVisor problemlos ausgetauscht werden können.

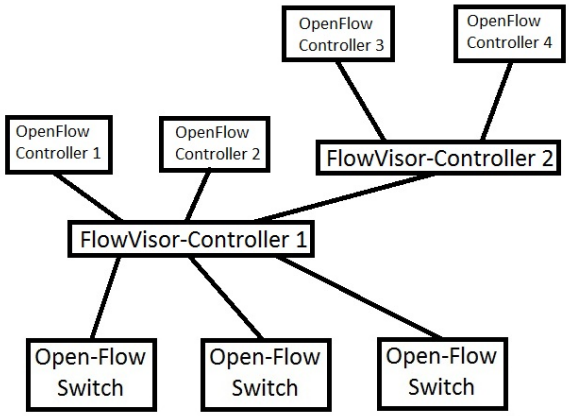


Abbildung 3. Beispiel eines Controller-Verbundes mittels FlowVisor

3. OpenFlow in der Praxis

3.1 Forschungsansätze

OpenFlow ermöglicht eine Vielzahl an Forschungsansätzen. So ließe sich zum Beispiel eine Alternative zum gängigen Routing-Verfahren testen. Wie oben beschrieben würden die neuen Routing-Regeln im Controller implementiert. Die Pakete würden von einem Computer des Forschers in das Netz geschickt, wo der Controller die Pakete erst nach den Regeln des experimentellen Routing-Protokolls als einen oder mehrere Flows definiert und dann alle Switches und Router innerhalb des Netzwerks entsprechend konfiguriert. Der Forscher kann dadurch sein Routing-Protokoll in einem großen Netzwerk testen und sein

Verhalten analysieren. Da man Flows beliebig deklarieren kann ließe sich auch eine Alternative zum IP-Protokoll realisieren. Dazu könnte ein Forschungs-Protokoll beispielsweise nur die MAC-Adressen aller Forschungsrechner im Netz berücksichtigen und eine eigenes System der Adressen-Vergabe auf der Vermittlungsschicht umsetzen. In der Praxis würden die Flows in diesem NON-IP-Netzwerk nur unter der Berücksichtigung der Ethernet-Header eines jeden Paketes definiert.

3.2 Anwendungsbeispiele

Neben der Forschung eröffnet OpenFlow auch noch ein breites Spektrum an zusätzlichen Anwendungsmöglichkeiten wie sie im Whitepaper zu OpenFlow [1] beschrieben werden. So könnte man beispielsweise den gesamten Traffic eines Netzwerks von dem Controller überwachen lassen, um Policies im Netzwerk durchzusetzen. Ein Controller könnte dabei alle VoIP-Pakete verwerfen lassen um diesen Service im OpenFlow-Netz zu sperren. Hierzu wäre dann auch nur ein Eintrag in den FlowTables aller Switches, Router und Access Points notwendig. Denkbar wäre auch ein Aufbau von VLANs mit Hilfe des OF-Protokolls. Die Switches würden dabei die von bestimmten Hosts versendeten Pakete mit VLAN-IDs versehen. So genannte Mobile wireless VoIP-Verbindungen ließen sich auch relativ einfach realisieren. Wenn ein Teilnehmer eines VoIP-Gesprächs den Access Point wechselt könnte der Datenstrom durch den Controller von alten zum neuen Access Point umgeleitet werden ohne, dass ein erneuter Verbindungsaufbau nötig wird. Somit wären die Teilnehmer in der Lage ihr Gespräch ohne Unterbrechung weiterführen. Ein weiteres im Whitepaper aufgeführtes Beispiel ist das Umleiten des Traffic durch ein offenes programmierbares System, wie etwa einen NetFPGA. Dort würde dann jedes Paket überprüft und ggf. modifiziert werden. Dadurch könnten dann verschiedenste Operationen, wie die Suche nach nicht autorisierten Eindringlingen oder das Konvertieren von Video-Streams durchgeführt werden. Im Folgenden werden drei, in ihrer Entwicklung weit fortgeschrittene Anwendungsbeispiele vorgestellt und beschrieben. Alle drei wurden auf der SIGCOMM, einer bedeutenden Konferenz für die Fachgebiete Kommunikation und Computer-Netzwerke, vorgestellt. Sie verdeutlichen noch einmal das große Potential von OpenFlow.

3.2.1 Plug-n-Serve

Plug-n-Serve ist ein dynamisches Lasten-Verteilungssystem auf OpenFlow-Basis. Der Controller steuert den Traffic zu und von den Servern innerhalb eines unstrukturierten Netzwerks. Der Traffic wird dabei nach den Regeln des eigens dafür entwickelten Optimierungs-Algorithmus LOBUS auf die Server verteilt. In der auf der SIGCOMM09 durchgeführten Demonstration und wie in Referenz [4] beschrieben wurde als Beispiel ein Netz von Webservern vorgeführt. Normalerweise würden durch ein statisches Lasten-Verteilungssystem die http-Anfragen gleichmäßig auf alle Server verteilt. Diese Methode kann für Netzwerke, die drauf ausgelegt sind viele Server möglichst gut an das Internet anzubinden, ausreichen, für den Einsatz in unstrukturierten Netzwerken wie Campus-Netzen ist sie allerdings ungeeignet. Plug-n-Serve verteilt im genannten Beispiel die http-Anfrage nicht gleichmäßig, sondern nach den Kriterien der Netz- und Serverauslastung. Die Routing-Entscheidung wird hierbei von dem im Controller implementierten LOBUS-Algorithmus getroffen. Wie in Abbildung 4 zu sehen besteht der Controller hier aus drei wesentlichen Komponenten, dem Net-Manager, der die Auslastung des Netzes überwacht, dem Host-Manager der Informationen über den Zustand der Server sammelt, und dem Flow-Manager der die Routing-Entscheidung für jede Anfrage fällt und dann entsprechend die FlowTables der Switches und Router konfiguriert. Der Vorteil dieses in Referenz [4] beschriebenen Systems ist eine effizientere Nutzung der vorhandenen Ressourcen und eine effektivere Anbindung der Server an das Internet.

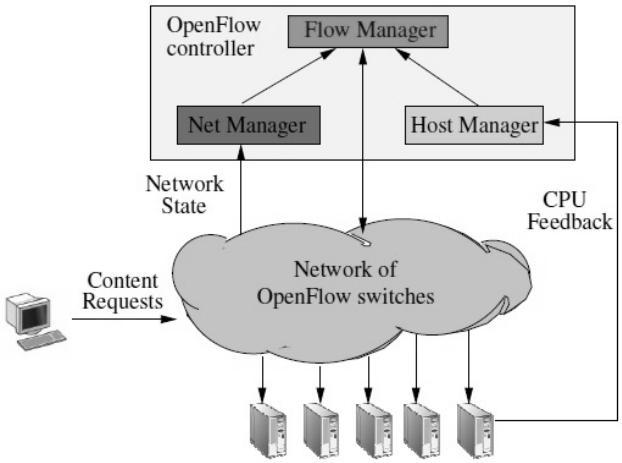


Abbildung 4. Ein Plug-n-Serve Netz [4]

3.2.2 OpenPipes

OpenPipes ist eine OpenFlow-Controller-Software zum Testen von High-Speed Networking Systems. Diese Systeme bestehen aus einer Reihe von Modulen die hintereinander oder parallel geschaltet und von einem Datenstrom durchlaufen werden. Dabei wird der Datenstrom in den einzelnen Modulen analysiert oder bearbeitet. Beispiele dafür sind bestimmte Firewalls oder Video-Bearbeitungs-Systeme. Das Problem bei der Entwicklung dieser Systeme ist, dass Design-Entscheidungen in der Entwurfsphase getroffen werden müssen und sich Tests unter realistischen Bedingungen schwierig gestalten. Referenz [5] bietet dazu mit OpenPipes folgende Lösung am Beispiel eines Video-Bearbeitungs-Systems. Wie in Abbildung 5 zu sehen werden die Video-Daten durch einen Host in ein Open-Flow-Netzwerk geschickt. Die Module befinden sich hierbei auf verschiedenen Computern im Netz und werden je nach Einstellung des Controllers durch verschiedene Module geleitet. Dabei lässt sich die Reihenfolge ändern um verschiedene Konfigurationen des Systems zu testen. Die Abbildung 5 zeigt die GUI der Controller-Software. Dabei symbolisieren die gestrichelten Rechtecke die Rechner im Netz, die Buchstaben die einzelnen Module und die Pfeile die Verbindungen, die OpenFlow zwischen ihnen herstellt. OpenPipes ist ein weiteres Beispiel dafür, was für vielfältige Anwendungsmöglichkeiten OpenFlow ermöglicht.

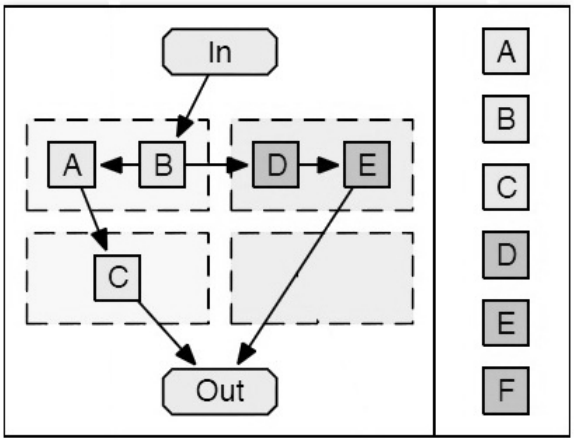


Abbildung 5. GUI der OpenPipes-Controller-Software [5]

3.2.3 Virtual Machine Mobility in OpenFlow

Unter Virtual Machine Mobility versteht man die Möglichkeit virtuelle Maschinen und deren Betriebssysteme im laufenden Betrieb von einem Rechner auf den anderen zu verlagern. So kann beispielsweise ein virtueller Spiele-Server auf einen anderen Computer verschoben werden ohne dass der Server neu gestartet werden muss. Das Problem hierbei ist allerdings, dass alle Clients, die eine Verbindung mit dem Server unterhalten, diese Verbindung verlieren, da der Server nach der Verlagerung nur mehr unter seiner neuen IP-Adresse zu erreichen ist. Die Referenz [6] beschreibt eine Lösung für dieses Problem. Auf der SIGCOMM08 wurde anhand dieses Beispiels demonstriert, wie OpenFlow es ermöglicht den Spiele-Server zu verschieben und gleichzeitig die Verbindung der Clients nicht abreißen zu lassen. Hierzu wurden mehrere OpenFlow-Netzwerke innerhalb der USA und Japan durch IP-Tunnel miteinander verbunden um ein größeres OpenFlow-Netz zu schaffen. An einem der Standorte lief der Spiele-Server, während die Clients von anderen Standorten aus zu dem Server eine Verbindung aufbauten. Auch hier wurde der virtuelle Spiele-Server auf einen anderen Rechner in einem anderen Netz verlagert. Die Verbindung zu den Clients blieb dabei allerdings erhalten. Dies ermöglichte der Controller indem er die Datenströme zwischen Server und den Clients auf die neue Position umlenkte. Die von den Clients versandten Pakete mit der nicht mehr aktuellen Zieladresse des Servers wurden vom Controller auf die neue Adresse umgeleitet. Diese Technik funktioniert auch wenn nicht der Server sondern der Client seinen Zugang zum Netzwerk wechselt, also sich zum Beispiel mit einem anderen Access Point verbindet. Dieser problemlose Positionswechsel der Netz-Teilnehmer eröffnet weitere interessante Möglichkeiten. So könnte ein Server durchgehend nach der Position im Netz suchen bei der er die optimale Verbindung zu seinen Clients schaffen kann um dann auf diese Position zu wechseln. Dieses Beispiel veranschaulicht noch einmal welches Potential in OpenFlow steckt und welchen Einfluss es auf die zukünftige Entwicklung von Netzwerken-Technologien haben kann.

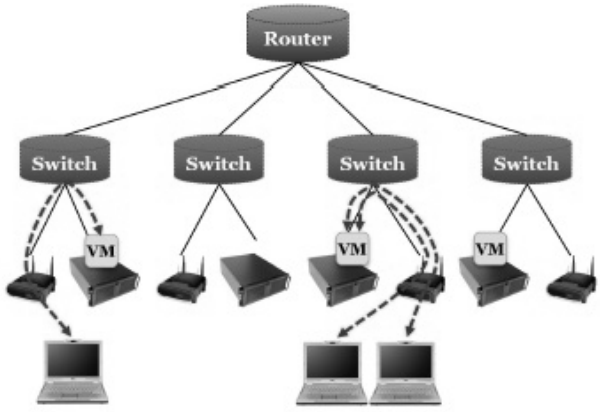


Abbildung 6. Ein OpenFlow-Netzwerk mit Virtuellen Maschinen [6]

4. Das OpenFlow Projekt

OpenFlow wurde an der Stanford Universität entwickelt. Mit der Entwicklung wurde 2007 begonnen und seitdem wird es durchgehend weiterentwickelt. Die Spezifikation [2] wurde 2009 in der Version 1.0.0 veröffentlicht. Das OpenFlow-Konsortium besteht größtenteils aus Forschern und Netzwerk-Administratoren. Jedoch kann jeder der die fachliche Kompetenz besitzt und sich einbringen will beitreten. Ihr Ziel ist es die Spezifikation zu verbessern und zu erweitern. OpenFlow ist für kommerzielle und nicht-kommerzielle Produkte frei nutzbar. Die Referenz [7] listet die wichtigsten Unterstützer der OpenFlow-Entwickler auf. Dazu zählen namhafte Firmen wie Cisco, HP, NEC, T-Mobile und Ericsson, die das OpenFlow-Projekt mit finanziellen Mitteln und die Bereitstellung von Hardware fördern. Viele Hersteller implementieren bereits seit mehreren Jahren das OpenFlow-Protokoll in ihre Geräte. Eine ganze Reihe an Forschungseinrichtungen und Universitäten konnten OpenFlow dadurch schon in ihre Netz-Infrastrukturen integrieren

5. ZUSAMMENFASSUNG

Obwohl OpenFlow erst vor wenigen Jahren entwickelt wurde kommt es schon verstärkt zum Einsatz. Die hier beschrieben Anwendungsbeispiele zeigen ein Teil der vielen Ideen die in OpenFlow-Netzen umgesetzt wurden und noch werden. Daher erscheint OpenFlow als eine vielversprechende Technologie. Das Magazin MIT Technology Review [7] zählt das OpenFlow-Protokoll zu einer wichtigen technologischen Neuentwicklung. Kritisch ist anzumerken, dass dieses Protokoll nicht ohne weiteres in Netzen einzusetzen ist, die die Größe von Universitäts-Netzwerken übersteigen, denn hier drängt sich die Frage auf wer die Macht besitzen darf den Controller zu kontrollieren. Der wohl größte Nachteil von OpenFlow ist allerdings die Tatsache, dass die in einem normalen Netzwerk eingesetzten Geräte durch OpenFlow-fähige Hardware ersetzt werden müssen, um überhaupt ein OpenFlow-Netzwerk aufbauen zu können. Nichtsdestoweniger wird die weitere Verbreitung von OpenFlow die Forschung auf dem Gebiet der Netzwerk-Technik unterstützen und positiv beeinflussen.

6. REFERENCES

[1] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S. und Turner, J.. OpenFlow: Enabling Innovation in Campus Networks, März 2008

[2] OpenFlow Switch Specification, Version1.0.0, Dez 2009, http://www.openflowswitch.org/wp/documents/

[3] Sherwood, R., Chan, M., Gibb, G., Handigol, N., Huang, T.-Y., Kazemian, P., Kobayashi, M., Underhill, D., Yap, K.-K., Appenzeller, G. und McKeown, N.. Carving Research Slices Out of Your Produciton Networks with OpenFlow, SIGCOMM 2009

[4] Handigol, N., Seetharaman, S., Flajslik, M., McKeown, N. und Ramesh, J., Plung-n-Serve: Load-Balancing Web Traffic using OpenFlow, SIGCOMM2009

[5] Gibb, G., Underhill, D., Covington, A., Yabe, T. und McKeown, N., OpenPipes: Prototyping high-speed networking systems, SIGCOMM2009

[6] Erickson, D., Gibb, G., Heller, B., Underhill, D., Naous, J., Appenzeller, G., Parulkar, G., McKewon, N., Rosenblum, M., Lam, M., Kumar, S., Alaria, V., Monclus, R., Bonomi, F., Tourrilhes, J., Yalagandula, P., Banerjee, S., Clark, C. und McGeer, R., A Demonstration of Virtual Machine Mobility in an OpenFlow Network, SIGCOMM 2008

[7] K. Greene. Special reports 10 emerging technologies 2009.MIT Technology Review, 2009. http://www.technologyreview.com/biotech/22120/

ISBN 3-937201-11-4

ISSN 1868-2634 (print)
ISSN 1868-2642 (electronic)