



Rico Herlt, B.Sc.

Cloudcomputing

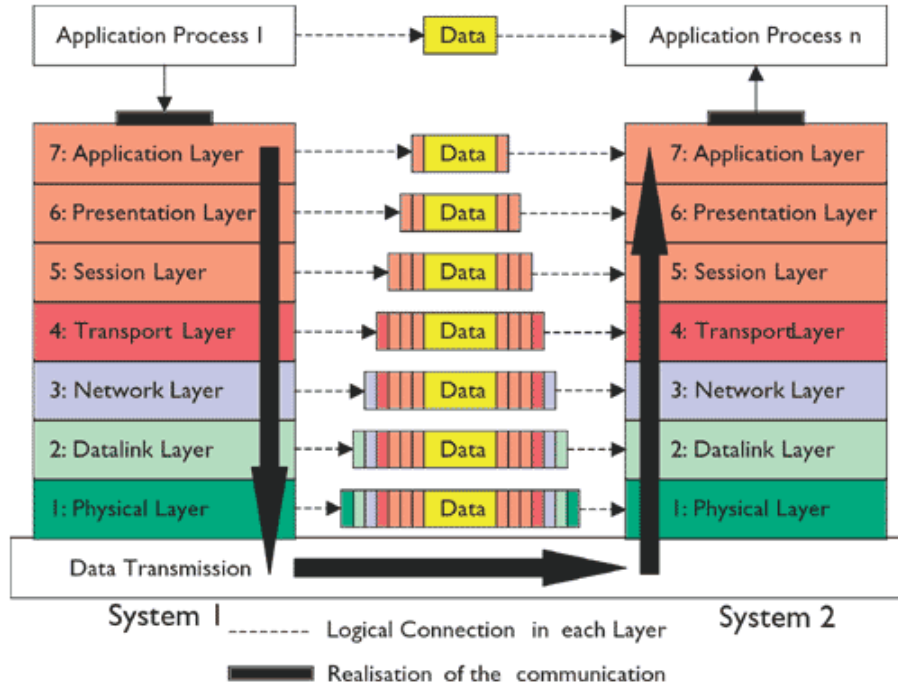
HTTP

02.12.16

Was ist HTTP?

- HTTP steht für **Hypertext Transfer Protocol**
- Zustandsloses Protokoll zur Übertragung von Daten auf Anwendungsschicht über ein Rechnernetz
- Hauptsächlich eingesetzt um Webseiten (oder andere Hypertext-Dokumente) aus dem WWW (World Wide Web) in einem Webbrowser zu laden, allerdings auch als allgemeines Dateiübertragungsprotokoll sehr verbreitet
- HTTP wurde von der Internet Engineering Task Force (IETF) und dem World Wide Web Consortium (W3C) standardisiert
- Aktuelle Version ist HTTP/2, spezifiziert in RFC 7540 (<https://tools.ietf.org/html/rfc7540>) RFC= Request for Comments)
- Es gibt Ergänzungen, wie zum Beispiel HTTPS zur Verschlüsselung übertragener Inhalte oder darauf basierende Standards, wie das Übertragungsprotokoll WebDAV (Web-based Distributed Authoring and Versioning) oder SOAP (ursprünglich für *Simple Object Access Protocol*)

OSI Referenzmodell



Bildquellen:

<https://techgimmick.files.wordpress.com/2015/07/ieb23protocol1.gif>

<https://www.techcress.com/wp-content/uploads/2015/01/OSI-Model-Burger.jpg>

OSI Referenzmodell

Layer 7 – Application Layer (Anwendungsschicht)

- Hier erfolgt die Eingabe und Ausgabe der Daten vom Benutzer
- Es wird die Software ausgeführt, die auf einem Betriebssystem läuft, egal ob Cloud-basiert oder Standalone.
- Diese Schicht stellt zum Beispiel Dienste für E-Mail, Telnet, Dateitransfer bereit
- Beispiel: Internetbrowser, FTP Client, Microsoft PowerPoint

Layer 6 – Presentation Layer (Präsentationsschicht)

- In der Präsentationsschicht wird das Betriebssystem ausgeführt
- Während die Benutzer mit der Anwendungsschicht interagieren, interagiert die Anwendungsschicht mit der Präsentationsschicht.
- Das geschieht direkt, oder mit einer speziellen Laufzeitumgebung, zum Beispiel der Java Runtime Environment (JRE)

OSI Referenzmodell

Layer 5 – Session Layer (Kommunikationssteuerungsschicht)

- Diese Schicht ist verantwortlich dafür, Sitzungen zwischen dem Betriebssystemen in der Präsentationssicht und anderen Maschinen zu verwalten.
- Zum Beispiel: Wenn jemand im Internet surft, interagiert er mit der Anwendungsschicht, diese wiederum mit der Präsentationsschicht und diese wiederum mit der Sitzungsschicht. Die Sitzungsschicht erlaubt dem Betriebssystem eine Verbindung mit dem Webserver herzustellen

Layer 4 – Transport Layer (Transportschicht)

- Verantwortlich für die Logistik der Sitzung
- Im vorherigen Beispiel wäre sie dafür verantwortlich herauszufinden welche und wie viele Informationen zwischen Betriebssystem und Web Server übertragen werden müssten

Layer 3 – Network Layer (Vermittlungsschicht)

- In dieser Schicht arbeiten Router
- Router sind Geräte, die Informationen paketweiser zwischen Computern weiter leiten.
- Im vorherigen Beispiel wären sie also dafür verantwortlich die Pakete zu senden und zu empfangen. Sender und Empfänger dieser Pakete kann eindeutig via IP-Adresse ermittelt werden.

OSI Referenzmodell

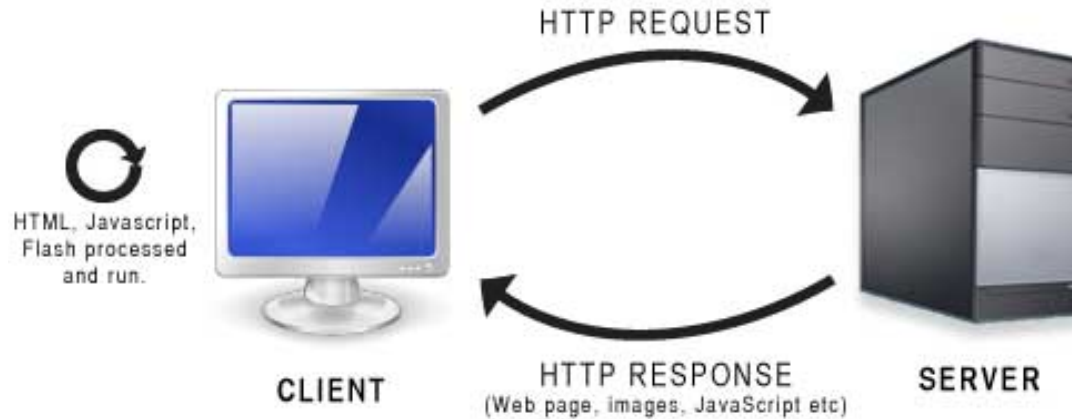
Layer 2 – Datalink Layer (Sicherungsschicht)

- Ebene, auf der Switches arbeiten und eine Verbindung zwischen zwei direkt verbundenen Netzknoten herstellen.
- Zudem werden hier Paketfehler erkannt und behoben, sofern möglich
- Es wird in zwei Bereiche unterteilt:
 1. Media Access Control (MAC): Verantwortlich wie Geräte mit dem Netzwerk verbunden sind, Bereitstellung des Zugriffs
 2. Logical Link Control (LLC): Überprüfen und beheben von möglichen Paketfehlern; Paketsynchronisierung

Layer 1 – Physical Layer (Bitübertragungsschicht)

- Sprichwörtlich die Hardware, die zum Netzwerk gehört, z.B. Kabel, Bluetooth, Brieftauben (Internet Protocol over Avian Carriers, https://de.wikipedia.org/wiki/Internet_Protocol_over_Avian_Carriers)
- Hauptaufgaben:
 - Physische Spezifikationen definieren
 - Protokolle definieren
 - Übertragungsmodi definieren (z.B. Halbduplex oder Vollduplex)
 - Netzwerktopologie definieren

Request- / Response-Prinzip

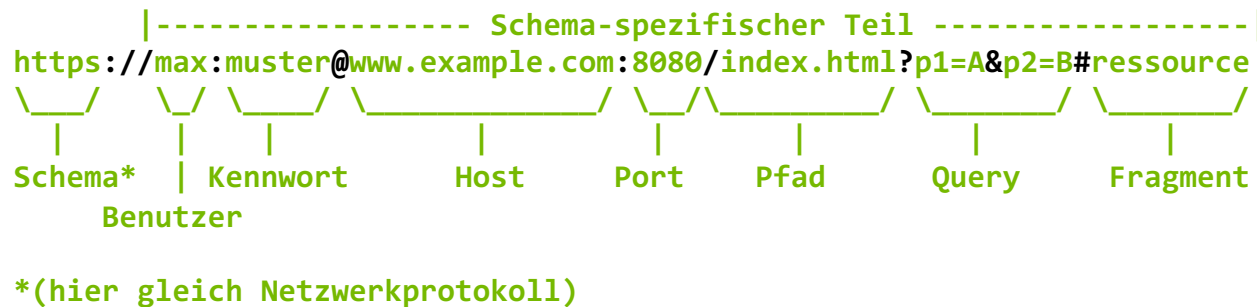


1. Client stellt eine Anfrage
2. Server antwortet

Bildquelle: <http://i.stack.imgur.com/eY5i4.jpg>

Uniform Resource Locator

- Ein Uniform Resource Locator (kurz **URL**) ist ein einheitlicher Ressourcenzeiger.
- Er identifiziert und lokalisiert eine Ressource und die damit zu verwendende Zugriffsmethode (Schema) auf die Ressource in Computernetzwerken
- Stellt eine Unterart der URI dar (Uniform Resource Identifier)
- Aufbau:



The diagram shows a URL `https://max:muster@www.example.com:8080/index.html?p1=A&p2=B#ressource` with brackets underneath identifying its components: `Schema*` (https), `Kennwort Benutzer` (max:muster), `Host` (www.example.com), `Port` (8080), `Pfad` (index.html), `Query` (p1=A&p2=B), and `Fragment` (#ressource). A dashed line above the first three parts labels them as the 'Schema-spezifischer Teil'. A note at the bottom states: `*(hier gleich Netzwerkprotokoll)`.

----- Schema-spezifischer Teil -----
`https://max:muster@www.example.com:8080/index.html?p1=A&p2=B#ressource`
Schema* Kennwort Benutzer Host Port Pfad Query Fragment
*(hier gleich Netzwerkprotokoll)

Aufbau einer HTTP Nachricht

Anfrage:

Verb

URL

Protokoll + Version

GET http://www.example.com/ HTTP/1.1

Accept: text/html, application/xhtml+xml, image/jxr, */*

Accept-Language: de-DE

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
[...gekürzt]

Accept-Encoding: gzip, deflate

Host: www.example.com

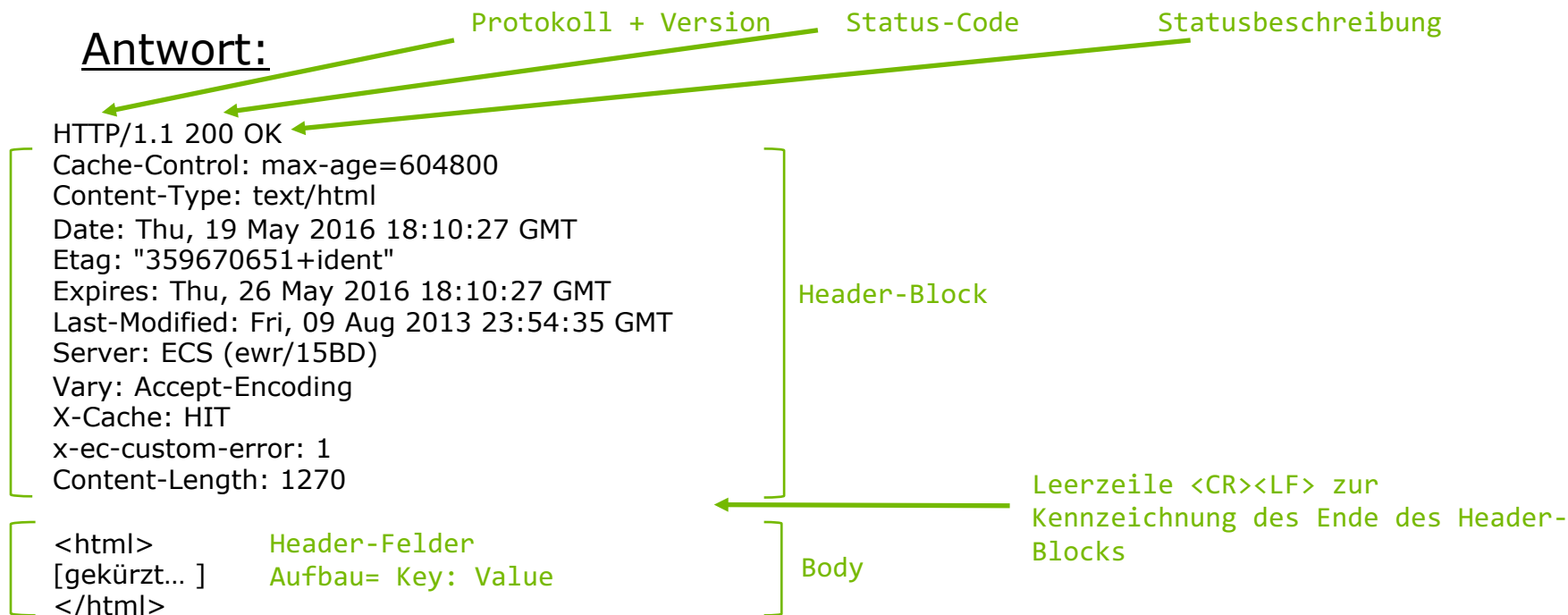
Connection: Keep-Alive

Header-Block

Header-Felder

Aufbau= Key: Value

Aufbau einer HTTP Nachricht



HTTP Verben

Es gibt eine ganze Reihe von HTTP-Verben. Die gebräuchlichsten und in der RFC 2616 (<https://www.ietf.org/rfc/rfc2616.txt>) spezifizierten Verben sind hier kurz erklärt:

Verb	Bedeutung
GET	Anfordern einer Ressource vom Server. Übertragung von begrenzten Daten in Form von Query-Parametern möglich.
HEAD	Weist den Server an, nur den Nachrichten-Header, jedoch nicht den Rumpf der Ressource zu übertragen
POST	Schickt unbegrenzte Mengen an Daten zu einem Server. Diese werden als Inhalt übertragen
PUT	Ähnlich wie POST, dient dazu eine Ressource zum Server hochzuladen. Es können sowohl Ressourcen verarbeitet als auch erstellt werden
DELETE	Löscht die angegebene Ressource auf dem Server
TRACE	Liefert die Anfrage so zurück, wie sie vom Server empfangen wurde
CONNECT	Stellt eine Verbindung zu einem Server her. Wird von Proxies implementiert, die SSL-Tunnel unterstützen
OPTIONS	Liefert eine Liste der vom Server unterstützten Methoden und Merkmale

HTTP Header

- Jeder HTTP Header-Block besteht aus vielen Header-Feldern.
- Jedes Feld ist nach dem Schlüssel-Wert-Prinzip (Key-Value) aufgebaut. Key und Value und wird durch einen Doppelpunkt. Jedes Schlüssel-Wert-Paar wird durch ein Zeilenumbruch terminiert
- Das Ende des letzten Header-Feld wird durch einen Doppelten Zeilenumbruch dargestellt. Dies kommt der Übermittlung von <CR><LF><CR><LF> gleich.
- Die meisten Header-Felder sind standardisiert
- Sie können jedoch projektspezifisch vom Entwickler erweitert werden
- Es wird zwischen Anfrage- (Request-) und Antwort- (Response-) Headern unterschieden
- Spezifiziert in RFC 2616 (<https://www.ietf.org/rfc/rfc2616.txt>)
- Übersichtliche Header-Felder-Liste (unvollständig) bei Wikipedia:
https://de.wikipedia.org/wiki/Liste_der_HTTP-Headerfelder

HTTP Header

Beispiele für Anfrage-Headerfelder:

Schlüssel	Wert (Beispiel)	Beschreibung
Accept	text/html	Inhaltstyp, der vom Client verarbeitet werden kann
Accept-Language	en-US	Sprache, die verarbeitet werden kann
Authorization	Basic dXNlcjpwYXM=	Authentifizierungstoken
Content-Length	348	Länge des Contents in Bytes
Content-Type	application/json	MIME-Typ des Bodies (POST- und PUT-Operationen)
If-Modified-Since	Sat, 29 Oct 1994 19:43:31 GMT	Erlaubt dem Server, den Statuscode <i>304 Not Modified</i> zu senden, falls sich seit dem angegebenen Zeitpunkt nichts verändert hat.
User-Agent	Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0)	Ermöglicht das Identifizieren des Clients, sodass z.B. bei Downloadseiten unterschiedliche Dateien für Windows und Mac angeboten werden können

HTTP Header

•Beispiele für Antwort-Headerfelder:

Schlüssel	Wert (Beispiel)	Beschreibung
Content-Type	text/html; charset=utf-8	MIME-Typ des Bodies
Date	Tue, 15 Nov 1994 08:12:31 GMT	Zeitpunkt des Absendens
Last-Modified	Tue, 15 Nov 1994 08:12:31 GMT	Zeitpunkt des letzten Änderns der Datei
Retry-After	120	alls eine Ressource zeitweise nicht verfügbar ist, so teilt der Server dem Client mit diesem Feld mit, wann sich ein neuer Versuch lohnt.
WWW-Authenticate	WWW-Authenticate: Basic	Definiert die Authentifikationsmethode, die genutzt werden soll, um eine bestimmte Datei herunterzuladen

HTTP Body

- Beinhaltet die eigentlichen Nutzdaten (Payload) der Nachricht.
- Typ des Nachrichtenkörpers wird durch das *Content-Type* Header-Feld spezifiziert
- Länge der Nachricht wird durch das *Content-Length* Header-Feld spezifiziert
- Inhalt kann sowohl binär, als auch in Textform, lesbar für das menschliche Auge, kodiert sein.
- HTTP-Verben mit Body in Anfragen (abgesehen bei Status *204 No Content*): POST, PUT, TRACE, OPTIONS, CONNECT
- HTTP-Verben ohne Body in Antworten: HEAD, DELETE

HTTP Status Codes

- Jede HTTP-Antwort beinhaltet einen Status Code und eine Statusbeschreibung.
- Spezifiziert in RFC 2616 (<https://www.ietf.org/rfc/rfc2616.txt>)
- Übersichtliche Liste auf Wikipedia: <https://de.wikipedia.org/wiki/HTTP-Statuscode>
- Status Codes lassen sich in folgende Gruppen unterteilen:

HTTP Status Codes

Gruppe	Typ	Beschreibung
100 - 199	Informationen	Die Bearbeitung der Anfrage dauert noch an.
200-299	Erfolgreiche Operationen	Die Anfrage war erfolgreich, die Antwort kann verwertet werden.
300-399	Umleitung	Um eine erfolgreiche Bearbeitung der Anfrage sicherzustellen, sind weitere Schritte seitens des Clients erforderlich.
400-499	Client-Fehler	Die Ursache des Scheiterns der Anfrage liegt im Verantwortungsbereich des Clients.
500-599	Server-Fehler	Nicht klar von den so genannten Client-Fehlern abzugrenzen. Die Ursache des Scheiterns der Anfrage liegt jedoch eher im Verantwortungsbereich des Servers.

HTTP Status Codes

Häufig vorkommende Status Codes:

Code	Beschreibung	Situation
102	Processing	Wird verwendet, um ein Timeout zu vermeiden, während der Server eine zeitintensive Anfrage bearbeitet
200	OK	Die Anfrage wurde erfolgreich bearbeitet.
201	Created	Die angeforderte Ressource wurde vor dem Senden der Antwort erstellt.
204	No Content	Erfolgreich verarbeitet, die Antwort enthält jedoch bewusst keine Daten.
301	Moved Permanently	Dauerhafte Umleitung
307	Temporary Redirect	Temporäre Umleitung
400	Bad Request	Anfrage-Nachricht war fehlerhaft aufgebaut

HTTP Status Codes

Häufig vorkommende Status Codes:

Code	Beschreibung	Situation
401	Unauthorized	Authentifizierung fehlt. Erforderte Art im „WWW-Authenticate“-Header-Feld spezifiziert
403	Forbidden	Die Anfrage wurde mangels Berechtigung des Clients nicht durchgeführt, bspw. weil der authentifizierte Benutzer nicht berechtigt ist, oder eine als HTTPS konfigurierte URL nur mit HTTP aufgerufen wurde.
404	Not Found	Die angeforderte Ressource wurde nicht gefunden.
500	Internal Server Error	Dies ist ein „Sammel-Statuscode“ für unerwartete Serverfehler.
501	Not Implemented	Die Funktionalität, um die Anfrage zu bearbeiten, wird von diesem Server nicht bereitgestellt.
503	Service Unavailable	Der Server steht temporär nicht zur Verfügung, zum Beispiel wegen Überlastung oder Wartungsarbeiten. Zeitpunkt erneuter Verfügbarkeit kann „Retry-After“-Header-Feld sein

Daten zum Server senden

Es gibt drei Möglichkeiten, um Daten vom Client zum Server zu senden:

1. Als benutzerdefiniertes Schlüssel-Wert-Paar im Header-Feld
2. Als Schlüssel-Wert-Paar im Query-Abschnitt in der URL:
 - Beispiel:
`http://mysite.com?key1=value1&key2=value2&key3=value3`
 - Bei allen HTTP-Verben möglich!
3. Im Nachrichtenkörper von POST- und PUT-Operationen

Demo: Manuelles HTTP

Folgende Programme eignen sich um HTTP-Nachrichten zu debuggen und manuell zu initiieren:

1. Fiddler (Windows): <http://www.telerik.com/fiddler>
 - Sehr komplexer Debugger
 - Bietet Performance
2. Postman (Mac + als Plugin im Chrome Browser bzw Chrome App):
<https://www.getpostman.com/>
 - Eignet sich um kurzerhand ohne großen Aufwand HTTP-Nachrichten zu versenden
 - Plattformunabhängig
3. Wireshark (für viele Plattformen): <https://www.wireshark.org/>
 - Sehr komplex
 - Unterstützt viele Protokolle, nicht nur HTTP



**Hochschule für Technik
und Wirtschaft Berlin**

University of Applied Sciences

www.htw-berlin.de