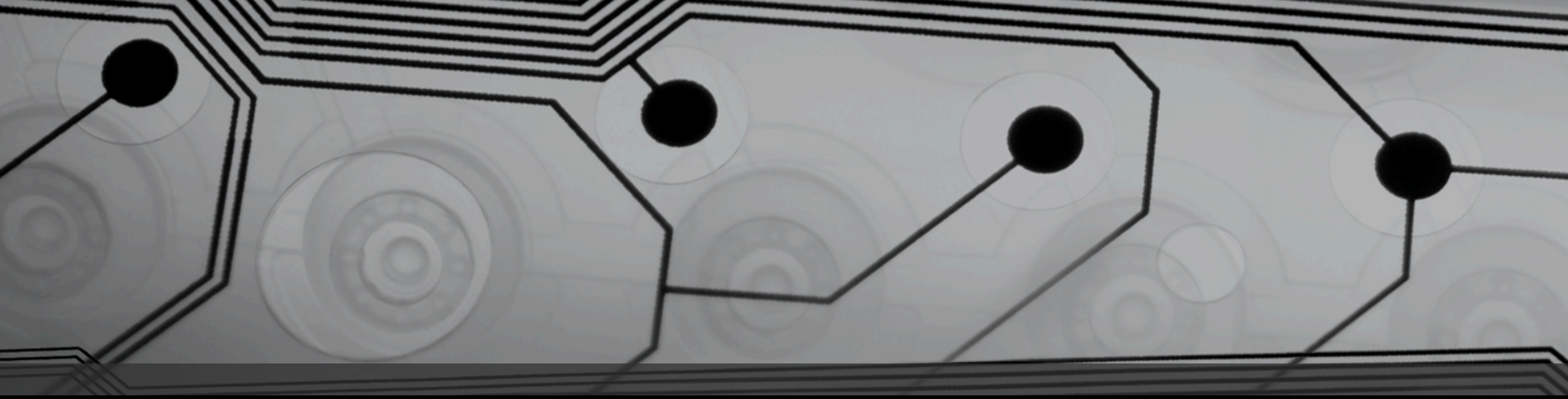


Hypertext Transfer Protocol

HTTP

Hatem Shugaa Addin

Übersicht

- Einleitung
- Aufbau
- Funktionsweise
- HTTP-Anfragensmethode
- HTTP-Statuscodes
- HTTP-Authentifizierung
- HTTP-Kompression
- HTTP-ABR (HTTP- Adaptive BIT Rate)
- HTTP – Versionen & HTTP(S)

Einleitung

- HTTP : Hypertext Transfer Protocol
- zustandloses Protokoll zur Übertragung von Datenn, vor allem Hypertext – Dokumenten (HTML, CSS, Skripte usw.), auf der Anwendungsschicht
- Entwickelt ab 1989 von Roy Fielding und Tim Berners Lee in CERN
- Wurde von IETF und W3C standardisiert
- Das berühmteste Internetprotokoll überhaupt!
- Es wird hauptsächlich eingesetzt, um Webseiten vom WWW in einem Webbrowser zu laden

Aufbau
(HTTP-Message)

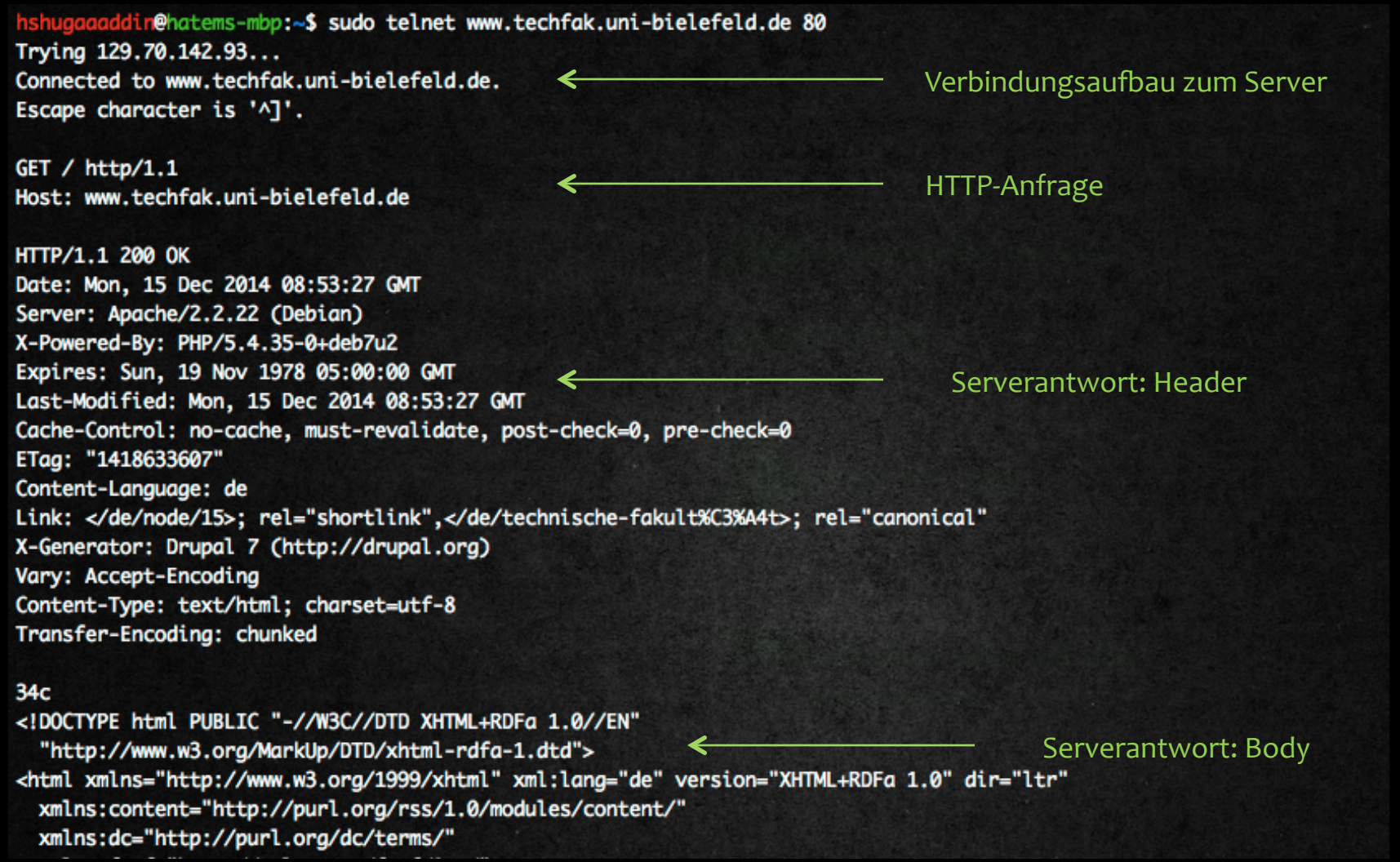
- Nachricht (message) : Kommunikation zwischen Client und Server

- Es gibt zwei Arten von Nachrichten
  - Anfrage (Request) vom Client an den Server
  - Antwort (Response) vom Server an den Client

- Bestandteile einer HTTP-Nachricht:
  - Message Header (HTTP-Header)
  - Message Body

Aufbau

(HTTP-Message)



Aufbau
(Inhalt vom Header)

- Header enthält Informationen über den Message-Body:
  - Kodierung (z.B utf-8)
  - Inhaltstyp (Text oder Bilder oder Video)

    → Für korrekte Interpretation vom Client

Aufbau
(HTTP-Headerfelder)

- Es gibt unterschiedliche HTTP-Headerfelder (Parameter & Argumente)
  - Allgemeine Header-Felder (in jeder HTTP-Nachricht)
    - Z.B Date
  - Request-Headerfelder
  - Response-Headerfelder
  - Entity-Headerfelder
    - Content-Type

- In RFC 2616 erklärt

Aufbau
(Request-Headerfelder)

- Accept: *Accept: text/html*
- Accept-Charset: *Accept-Charset: utf-8*
- Accept-Encoding: *Accept-Encoding: gzip, deflate*
- Accept-Language: *Accept-Language: en-US*
- Content-Length (in Byte) *Content-Length: 255*
- Date *Date: Tue, 16 Dec 2014 12:20:32 GMT*
- From *From: nutzer@beispiel.de*
- Host *Host: uni-bielefeld.de*
- usw…. Siehe RFC 2616

Aufbau
(Response-Headerfelder)

- Accept-Ranges: *Accept-Ranges: bytes*
- Age: *Age: 16*
- Allow: *Allow: GET, HEAD*
- Cache-Control: *Cache-Control: max-age=3600*
- Content-Language: *Content-Language: en-US*
- Content-Length *Content-Length: 255*
- Content-Location: *Content-Location: /pingo.html.de*
- Content-Type: *Content-Type: text/html; charset=utf-8*
- Proxy-Authenticate: *Proxy-Authenticate: Basic*
- usw…. Siehe RFC 2616

Aufbau
(Inhalt vom Body)

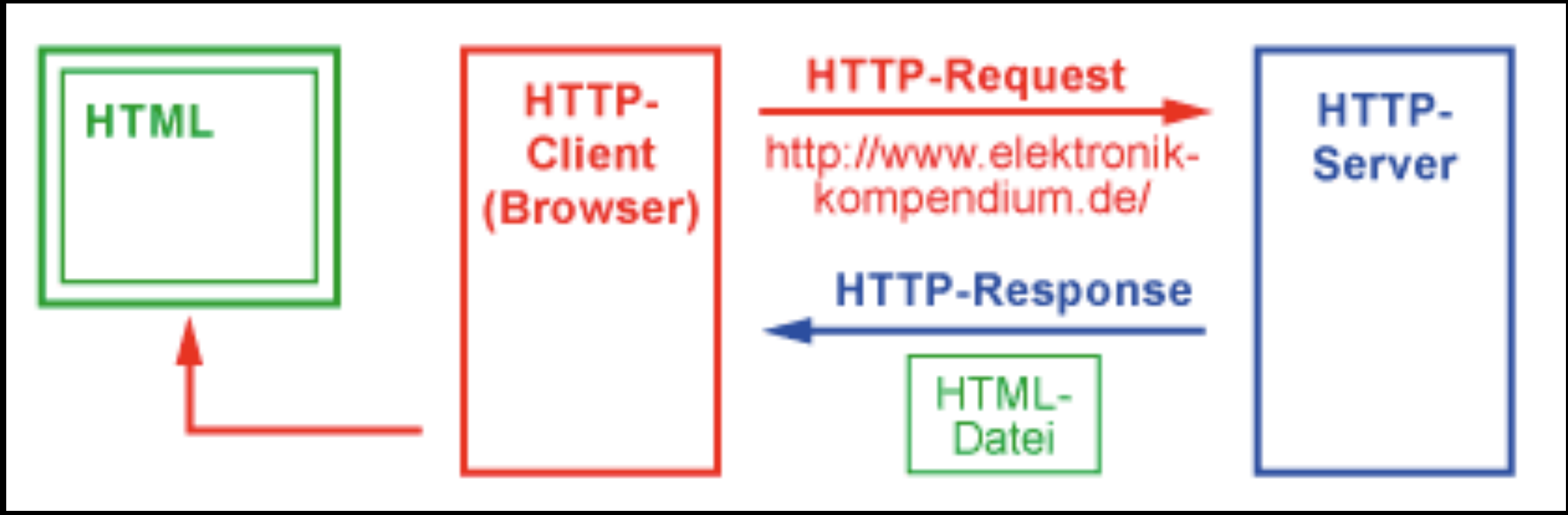
- Message-Body enthält schließlich die Nutzdaten

- z.B :
  - HTML
  - XML
  - CSS
  - Javascript
  - Bilder
  - Beliebige Dateien in beliebigen Typen

Funktionsweise

- Ziel: die Seite http://www.beispiel.de/text.html zu erreichen
  - Auf diese URL klicken
    - Der Server namens www.beispiel.de bekommt eine Anfrage (Request)
    - Client (webbrowser) möchte die Ressource /text.html erhalten
    - Der Name www.beispiel.de wird zuerst über ein DNS-Protokoll in die IP-Adresse umgesetzt
    - HTTP-GET-Anforderung wird über TCP auf den Standard Port 80 des HTTP-Server gesendet:
      - Anfrage: GET /text.html
Host: www.beispiel.de
    - Zusätzliche Informationen (Sprache) können über den HTTP-Header übertragen werden
    - Nach einer Leerezeile wird eine Antwort (Response) vom Server gesendet
    - Response besteht aus
      - Header-Informationen des Servers
      - Inhalt der Nachricht, hier Inhalt von text.html

Funktionsweise



Funktionsweise (response)

- Wie sieht eine Antwort aus?

HTTP/ 1.1 200 OK

Server: Apache/1.3.29 (Unix) PHP/4.3.4

Content-Length: 16122014

Content-Language: de

Content-Type: text/html

……..

(Inhalt von text.html)

HTTP-Anfragenmethode

- Jede Request wird durch die Angabe einer Methode eingeleitet
- HTTP-Methoden (groß geschrieben!):
  - GET
  - POST
  - HEAD
  - PUT
  - DELETE
  - TRACE
  - OPTION
  - CONNECT

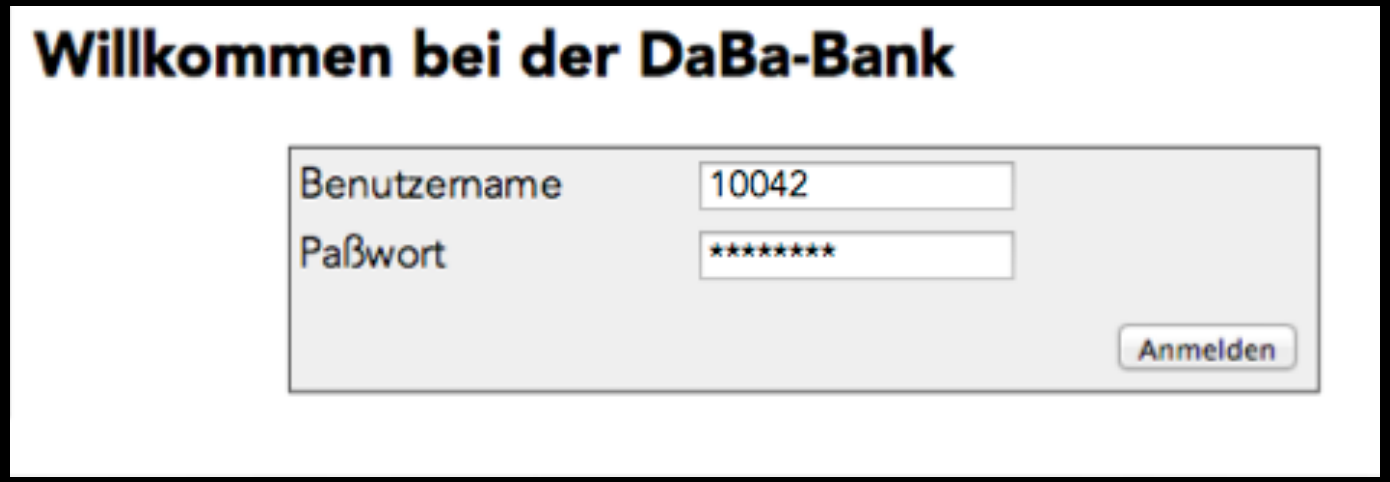
HTTP-Anfragenmethode
(GET-Methode)

- Die gebräuchlichste Methode

- Ressource unter Angabe einer URL vom Server anfordern

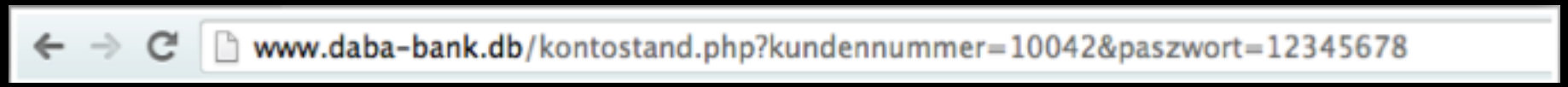
- Übermittlung von Formular-Daten
  - Daten werden in kodierter Form der URL angehängt
  - Daten mit ? voneinander getrennt

HTTP-Anfragenmethode
(GET-Methode)



HTTP-Anfragenmethode
(GET-Methode)

```
GET /kontostand.php?kundennummer=10042&paszwort=12345678 HTTP/1.1
Host: www.daba-bank.db
```



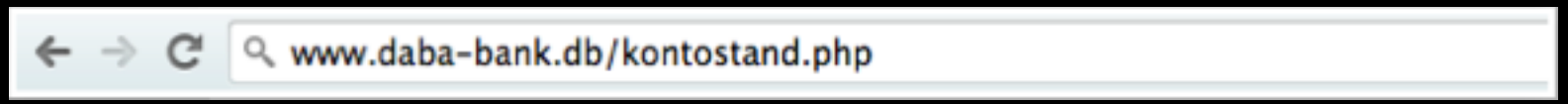
HTTP-Anfragenmethode
(POST-Methode)

- ähnlich wie GET-Methode
- Übergabe im Body der Anfrage
- Übermittlung von Formular-Daten

HTTP-Anfragenmethode
(POST-Methode)

```
POST /kontostand.php HTTP/1.1
Host: www.daba-bank.db
Content-Type: application/x-www-form-urlencoded
Content-Length: 36

kundennummer=10042&paszwort=12345678
```



HTTP-Anfragenmethode
(PUT-Methode)

- Erstellen (hochladen) und Ändern (modifizieren) von Daten

- Angabe der Ziel-URL auf dem Server

HTTP-Anfragenmethode
(DELETE, TRACE, OPTION-Methoden)

- DELETE:
  - Löschen einer Datei, die mit einer URL adressiert ist
- TRACE:
  - Dient der Verfolgung von HTTP-Requests
  - Sinnvoll für Debugging von Verbindungen
- OPTION:
  - Liste der vom Server unterstützten Methoden
  - Multiple choice

HTTP-Statuscodes

- Jede HTTP-Anfrage wird vom Server mit einem HTTP-Statuscode beantwortet
- Gibt Informationen dadrüber, ob die Anfrage erfolgreich bearbeitet wurde
- Im Fehlerfall wird Client (Browser) informiert

HTTP-Statuscodes
(1xx -Informationen)

- 1xx –Informationen:
  - Zwischenantwort
  - Die Anfrage ist noch in Bearbeitung
  - Nützlich, wegen Zeitüberschreitungsgrenze des Clients

HTTP-Statuscodes
(2xx –Erfolgreiche Operation)

- 2xx – Erfolgreiche Operation
  - Meldung, dass die Anfrage erfolgreich vom Server bearbeitet wurde
  - Eine Antwort wird an den Client zurückgesendet
  - Alles GUT ☺

HTTP-Statuscodes
(3xx –Umleitung)

- 3xx – Umleitung
  - Mehrere Schritte seitens des Clients sind manchmal erforderlich
  - Gibt dem Client Bescheid, nach welchem Ziel eine neue Anfrage hergestellt werden sollte
  - Die Daten sind nicht immer am gleichen Platz!

HTTP-Statuscodes
(4xx –Client-Fehler)

- 4xx – Client-Fehler
  - Auftritt eines Fehlers bei der Bearbeitung einer Anfrage
  - Der Fehler liegt im Verantwortungsbereich des Clients
  - z.B: eine Anfrage, die vom Server nicht bearbeitet werden kann
  - Die häufigsten HTTP-Statuscodes

HTTP-Statuscodes
(5xx –Server-Fehler)

- 5xx – Server-Fehler
  - Auftritt eines Fehler bei der Bearbeitung einer Anfrage
  - Der Fehler ist im Server

Status-Code	Meldung EN	Meldung DE	HTTP-Version
100	Continue	Weiterleitung	1.1
101	Switching Protocols	Protokoll ändern	1.1
200	OK	Alles in Ordnung / Standardmeldung	1.0
201	Created	Erfolgreich erstellt	1.0
202	Accepted	Akzeptiert	1.0
203	Non-Authoritative Information	Die Daten sind von einem anderen Server	1.0
204	No Content	Ausführung ohne Feedback	1.0
205	Reset Content	Datenversendung nicht möglich	1.1
206	Partial Content	Datenversendung in mehreren Teilen	1.1
300	Multiple Choices	Datei ist mehrfach vorhanden	1.0
301	Moves Permanently	Datei ist dauerhaft verschoben	1.0
302	Moves Temporarily	Datei ist vorübergehend verschoben	1.0
303	See Other	Datei ist an einem andern Ort	1.0
304	Not Modified	Datei ist unverändert	1.0
305	Use Proxy	Datei über Proxy anfordern	1.0
400	Bad Request	Keine Bearbeitung	1.0
401	Unauthorized	Nicht autorisiert	1.0
402	Payment Required	Kostenpflichtige Daten	1.0
403	Forbidden	Zugriff verweigert	1.0
404	Not Found	Nicht gefunden	1.0
405	Method not Allowed	Methode nicht erlaubt	1.1
406	Not Acceptable	Anfrage nicht akzeptiert	1.1
407	Proy Authorisation Required	Nicht autorisierter Proxy	1.1
408	Request Timeout	Zeitüberschreitung	1.1
409	Conflict	Zugriff gesperrt	1.1
410	Gone	Daten wurden verschoben	1.1
411	Length Required	Längenangaben erforderlich	1.1
412	Precondition Failed	Vorbedingungen nicht erfüllt	1.1
413	Request Entity Too Long	Anfrage zur Bearbeitung zu lang	1.1
414	Request URL Too Long	Adresse zu lang	1.1
415	Unsupported Media Type	Nicht unterstützter MIME-Type	1.1
416	Requested Range Not Satisfiable	Angegebener Bereich ist falsch	1.1
417	Expectation Failed	Vorraussetzung ist nicht erfüllt	1.1
500	Internal Server Error	Interner Server Fehler	1.0
501	Not Implemented	Nicht implementierte Anfrage	1.0
502	Bad Gateway	Gatewayfehler	1.0
503	Service Unavailable	Server ueberlastet	1.0
504	Gateway Timeout	Zeitüberschreitung Gateway	1.1
505	HTTP Version not supported	Nicht unterstützte HTTP-Version	1.1

HTTP-Authentifizierung

- Zugriff nur für begrenzten Personenkreis
- Frage nach Benutzernamen und Passwort
- Bei falschem Benutzername bzw. Passwort:
  - 401 –Statuscode “Unauthorized”

- HTTP-Authentifizierung:
  - Basic Authentication
  - Digest Access Authentication
- In RFC 2617

HTTP-Authentifizierung
(Basic Authentication)

- Häufigste Art der HTTP-Authentifizierung
- Server fordert Authentifizierung an
- Client zeigt Dialog für Benutzername- und Passworteingabe
- Client sendet die Auth. mit Authorization-Header in Form eines in Base64 kodierten Textes an den Server
- Sichere Übertragung?
  - ÜBERHAUPT NICHT!

HTTP-Authentifizierung
(Basic Authentication: Base64)

- Warum nicht sicher?
  - Base64!

Data Bytes in Decimal Form	**212**		**39**	**247**
Data Bytes In Binary Form	11010100		00100111	11110111
Data Rearranged Into 6-Bit Groups	110101	000010	011111	110111
6-Bit Groups In Decimal Form	53	2	31	55
Groups Converted To ASCII Characters	**1**	**C**	**f**	**3**

Im Header:

Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==

HTTP-Authentifizierung
(Digest Access Authentication)

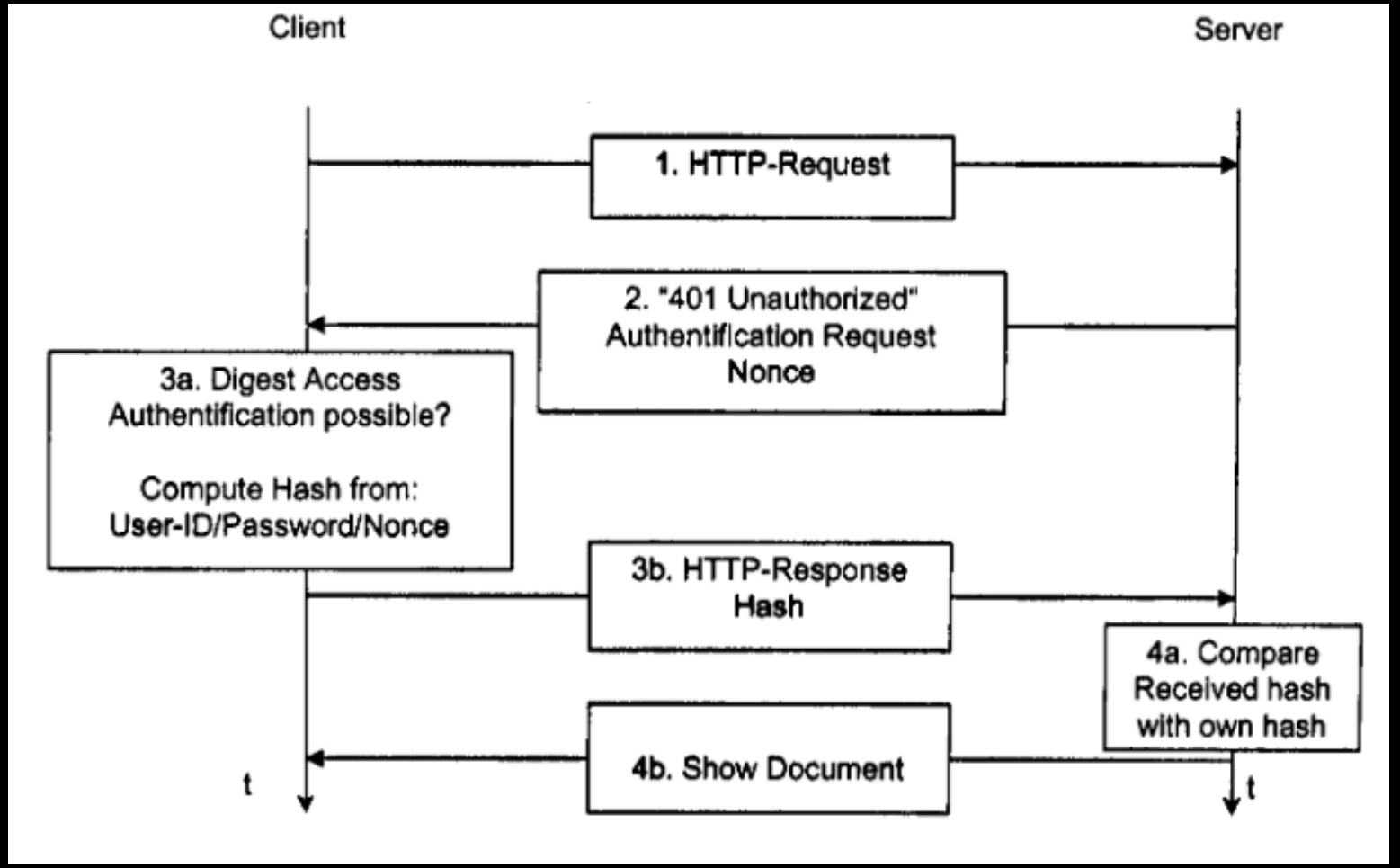
- Sichere Variante
- Server sendet zusätzlich eine eigens erzeugte zufällige Zeichenfolge (Nonce)
- Nonce: used only once
- Client berechnet den Hashcode (MD5)
- Hashcode von den gesamten Daten:
  - Benutzername + Passwort + Nonce + HTTP-Methode + angeforderte URI
- Hashcode im Authorization-Header an den Server zurücksenden
- Server berechnet den Hashcode
- Vergleich: Server-Hashcode == Client-Hashcode?

HTTP-Authentifizierung
(Digest Access Authentication:Hashcode)

- MD5:
  - One-way Codierung
  - Sehr schwer zu entziffern
  - Nur Ausgabe vorhanden

- Digest Access Authentication:
  - MD5(Benutzername + Passwort + URI + Nonce) = Hashcode

HTTP-Authentifizierung
(Digest Access Authentication)



HTTP-Kompression

- Ziel:
  - Übertragene Datenmenge verringern
  - Schnelle Übertragung

- HTTP-Server Antworten komprimieren!
- Client muss seine Kompressionsfähigkeiten dem Server mitteilen
- Client-Mitteilung durch Header-Feld Accept-Encoding (gzip, deflate)
- Server-Antwort mit Header-Feld Content-Encoding
- Sehr sinnvoll bei textuellen Daten (HTML, CSS, Javascript)

HTTP-ABR (HTTP Adaptive Bit Rate)

- Verfahren für Übertragung von Videostreaming per HTTP

- Anforderung bei der Übertragung:
  - Bildschirmgröße
  - Bandbreite von 100 KBit/s bis 100 Mbit/s

- HTTP-Server müsste unterschiedliche Qualitäts- und Geschwindigkeitsanforderung berücksichtigen
- Anpassung der Übertragungsrate an die verfügbare Bandbreite
- Alternativen zu HTTP-ABR
  - HLS von Apple
  - HSS von Microsoft
  - HDS von Adobe

HTTP-Versionen & HTTPS
(HTTP/0.9 & HTTP/1.0)

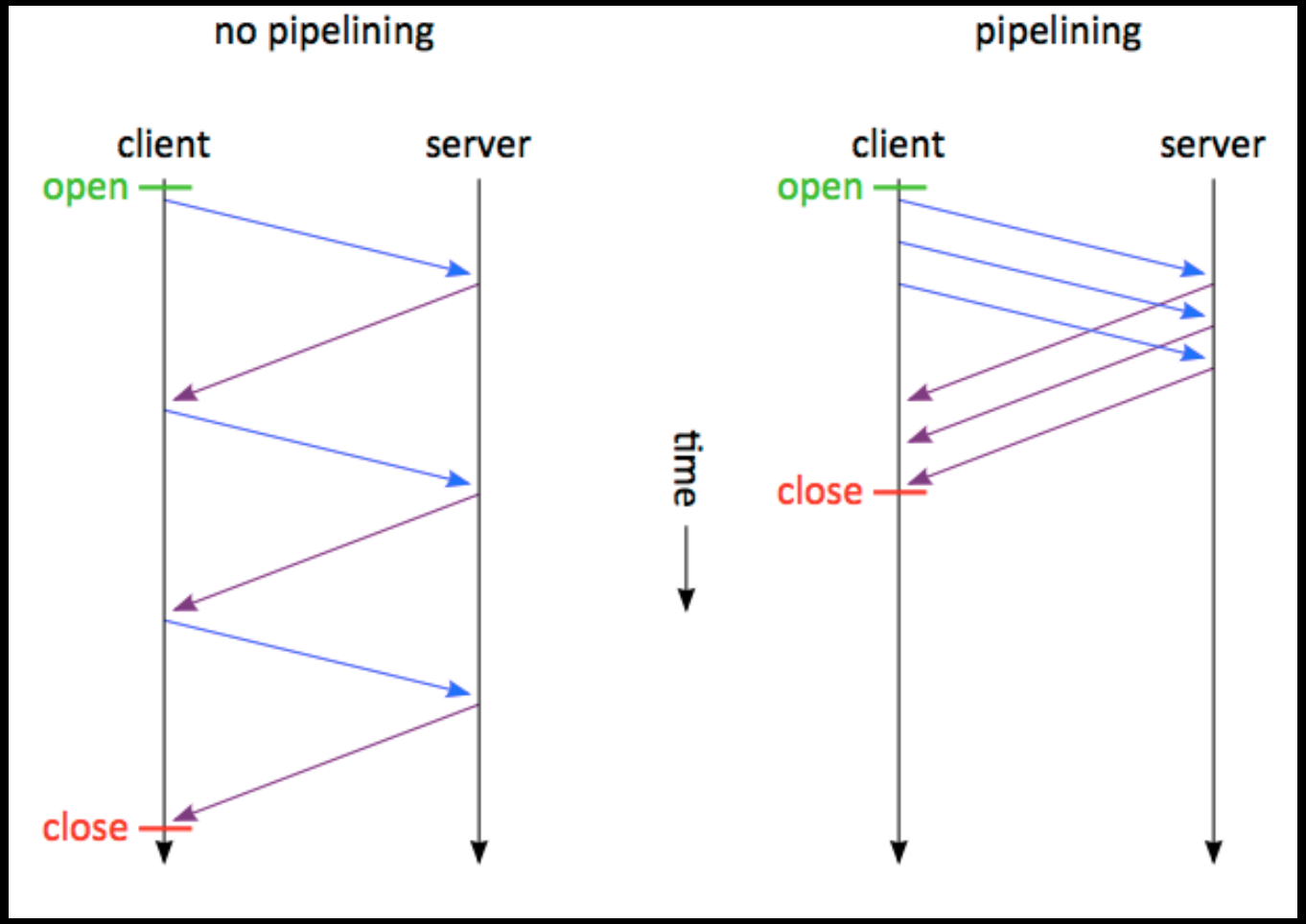
- 1991 : erste HTTP-Version 0.9

- 1996: HTTP/1.0
  - Vor jeder Anfrage eine neue TCP-Verbindung
  - Nach der Übertragung der Antwort wird TCP-Verb. vom Server wieder geschlossen
  - Wenn die Antwort sehr groß ist?
    - Mehrere TCP-Verbindungen

HTTP-Versionen & HTTPS
(HTTP/1.1)

- 1999: HTTP/1.1
- Zusätzliche Header-Feld namens keepalive
    - Keinen Verbindungsabbau (persistent connection)
- Abgebrochene Übertragungen können fortgesetzt werden
- Bei einer großen Antwort nur ein TCP-Verbindung (HTTP-Pipelining)
    - SCHNELLER!

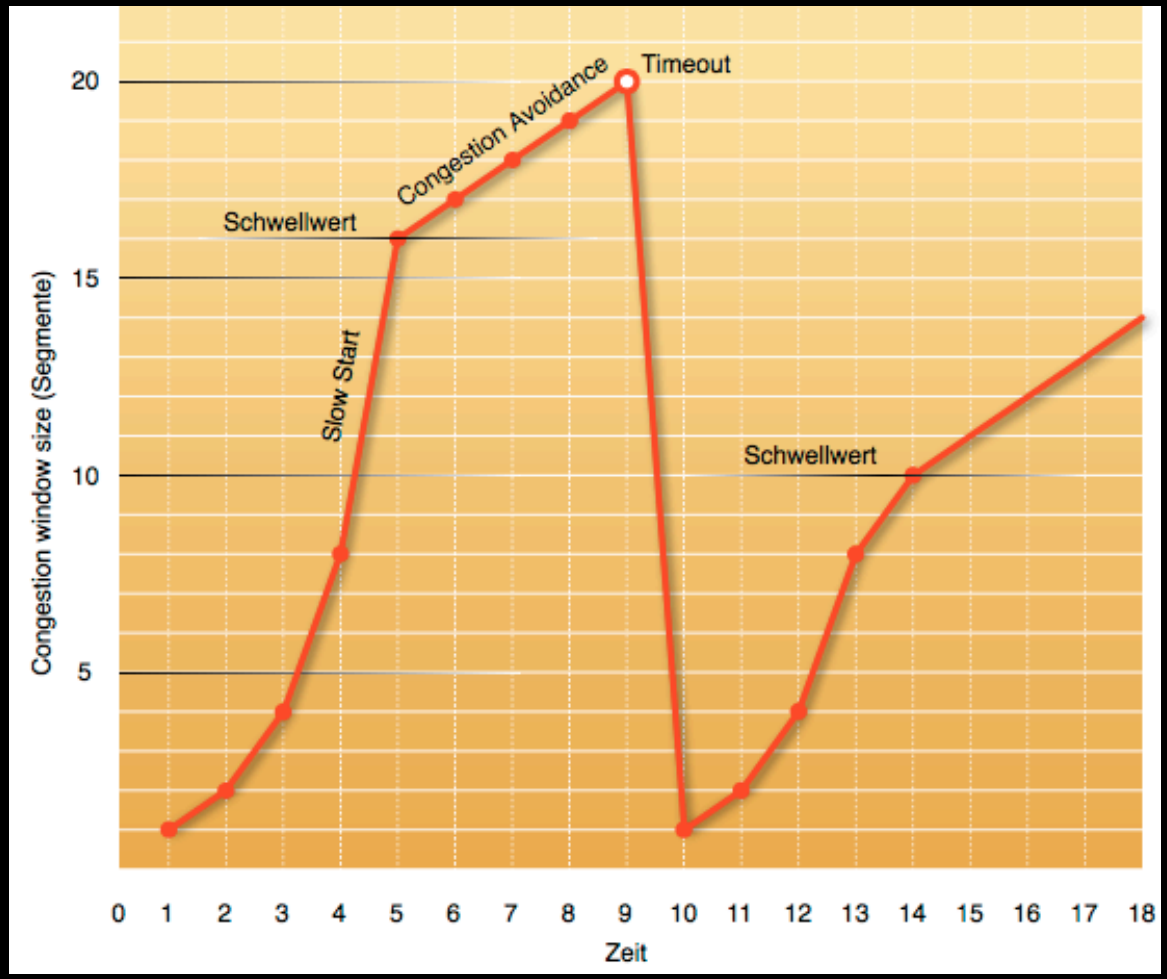
HTTP-Versionen & HTTPS
(HTTP-Pipelining)



HTTP-Versionen & HTTPS
(HTTP/1.1)

- Warum ist die Geschwindigkeit der Übertragung bei mehreren TCP-Verbindung langsam?
  - Wegen slow-start-Algorithmus bei TCP-Verbindungen
  - Mehrere TCP-Verbindungen bedeuten mehrere slow-start-Abläufe

HTTP-Versionen & HTTPS
(HTTP/1.1)

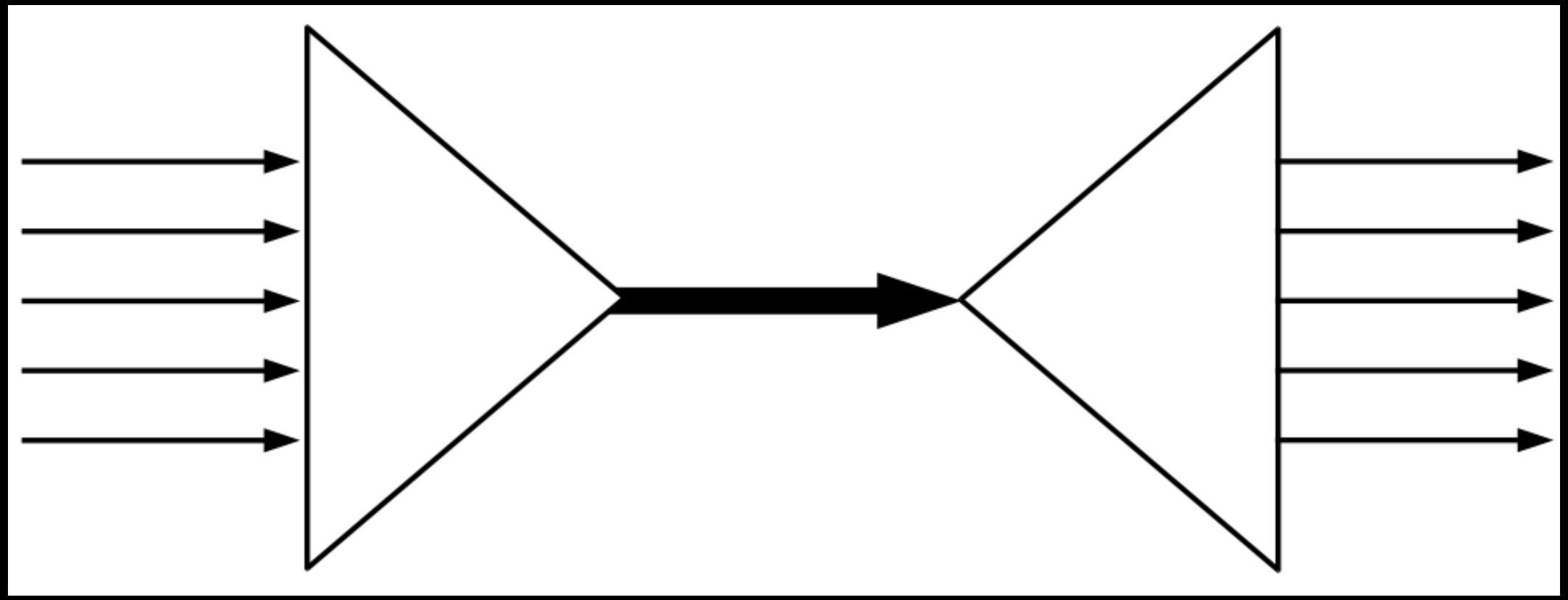


HTTP-Versionen & HTTPS (HTTP/2)

- Wo ist HTTP/2?
  - In der Entwicklungsphase
- Vorschläge:
  - Von Microsoft : HTTP Speed + Mobility
  - Von Google: SPDY (speedy)
  - Schon teilweise in den Browsern von Google und Microsoft umgesetzt
- Anforderung an HTTP/2
  - Abwärtskompatibel zu HTTP/1.1
  - Weitergehende Daten-Kompressionsmöglichkeiten durch HPACK-Algorithmus
  - Beschleunigung und Optimierung der Übertragung durch
    - Zusammenfassen mehrere Anfragen (Multiplex-Verfahren) in einer Verbindung

HTTP-Versionen & HTTPS

(HTTP/2 – Multiplex-Verbindung)



HTTP-Versionen & HTTPS (HTTP/3)

- HTTP/2 nicht optimal
- Mängel von HTTP/2 möglichst zeitnah beheben
- In Entwicklungsphase auch!

HTTP-Versionen & HTTPS
(HTTPS – RFC 2818)

- HTTP ist unsicher!
- Lösung: HTTPS (Hypertext transfer Protocol Secure)
- ALLES wird verschlüsselt übermittelt
- Sicherheitsschicht zwischen Anwendungsschicht und Transportschicht
- Verwendet SSL bzw. TLS

Vielen Dank für Ihre Aufmerksamkeit

Quellen

- http://www.elektronik-kompendium.de/sites/net/0902231.htm
- https://unimoodle.uni-bielefeld.de/pluginfile.php/18833/mod_resource/content/2/Datenbankanwendungen_2014.pdf
- http://de.wikipedia.org/wiki/Hypertext_Transfer_Protocol_Secure
- http://de.wikipedia.org/wiki/Liste_der_HTTP-Headerfelder
- http://de.wikipedia.org/wiki/HTTP-Pipelining
- http://www.codeproject.com/Articles/162726/Digest-Authentication-on-a-WCF-REST-Service
- http://de.wikipedia.org/wiki/HTTP-Statuscode
- http://aktuell.de.selfhtml.org/artikel/programmiertechnik/useronline/ablauf_http_kommunikation.png
- http://www.tcpipguide.com/free/diagrams/mimebase64.png
- https://en.wikipedia.org/wiki/Multiplexing