Vorlesungsreihe
Entwicklung webbasierter Anwendungen

Protokolle auf der Anwendungsschicht
smtp / pop3 / ftp / http

Prof. Dr.-Ing. Thomas Wiedemann
email: wiedem@informatik.htw-dresden.de

HOCHSCHULE FÜR TECHNIK UND WIRTSCHAFT DRESDEN (FH)
Fachbereich Informatik/Mathematik

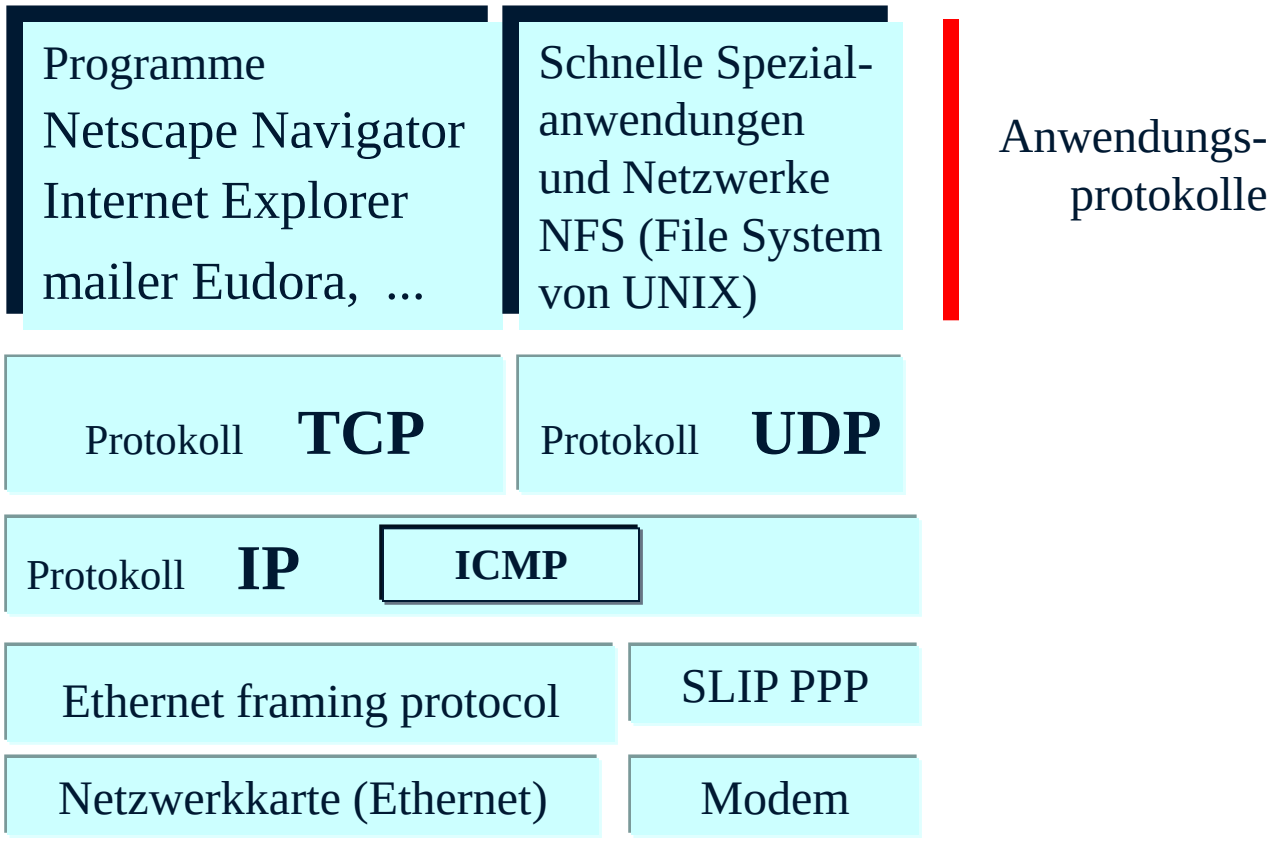
Gliederung

- **Einordnung der Anwendungsprotokolle**
- **Prinzipielle Architekturen**
- **Emailprotokolle smtp / pop3 / IMAP**
- **Datenübertragungsprotokoll FTP**
- **Hypertext Transfer Protocol HTTP**

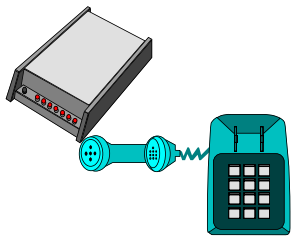
Einordnung der Anwendungsprotokolle

OSI-Ebene	OSI-allgemein
7	Application
6	Presentation
5	Session
4	Transport
3	Network
2	Logical
1	Physical

Am Beispiel der Internet-Ebenen



Die Anwendungsprotokolle setzen auf den Basisprotokollen TCP/IP oder UDP auf und decken die OSI-Schichten 5-7 ab.



Internet-Anwendungsprotokolle

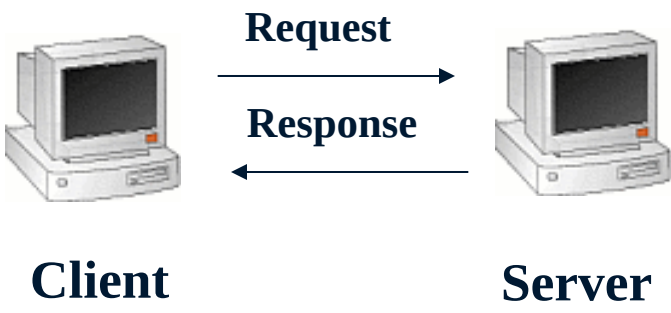
- definieren die Kommunikation zwischen Anwendungen
- in der Regel Client/Server-basiert
- setzen grundsätzlich auf TCP oder UDP auf
- es können mehrere Server eines Typ auf einem Rechner laufen
- Unterscheidung nach Portnummern, z.B.:

  www.test.de - 1. Standard-Webserver (auf Standardport 80)

  www.test.de:88 - 2. Webserver auf Port 88

Grundlegende Architektur und Kommunikation

- Server ist im "Horchmodus" auf bestimmten Port
- Client sendet Request an Serverport
- Server startet Bearbeitungsprozeß und geht wieder in Horchmodus
- Bearbeitungsprozeß bearbeitet Anfrage und sendet Antwort



Protokolle zum Emailversand

Historie und Ausblick

- Erster Email-Versand im Herbst 1971 durch BBN-Techniker Ray Tomlinson zwischen zwei Rechnern über das Arpanet
- Bei Suche nach Adreßtrennzeichen verwendete er das bis dahin selten genutzte @-Zeichen
- Programmierung der Software Sendmail im Jahr 1981 durch Eric Allman – diese ermöglichte einen Versand gleichzeitig in verschiedene Netze
- Die E-Mail-Kommunikation basiert auf drei Protokollen:
  - SMTP zum Versenden
  - POP und IMAP zum Empfangen von Nachrichten
  - Spezifikationen sind in entsprechenden RFCs festgelegt.
- Auch gegenwärtig wird trotz neuer Alternativen wie Instant Messenging die E-Mail immer beliebter :
  - Im Jahr 2005 gab es rund 36 Milliarden versandte E-Mails pro Tag
  - Heute (Stand 2015) werden dienstlich/privat jeweils 120 Mrd.,also in der Summe ca. 250 Mrd. Email weltweit versendet (Quelle: http://de.statista.com/statistik/daten/studie/247225/umfrage/schaetzung-zur-zahl-der-taeglich-verschickten-e-mails/)
  - trotz SPAM sind Emails eine attraktive Option, da Kundenkontakt im PUSH-Modus

Quelle(n) : www.tecchannel.de

SMTP - Simple Mail Transfer Protocol

- dient zum zuverlässigen und effiziente Transport von Nachrichten. SMTP ist unabhängig vom Netzprotokoll
- meist wird Port 25 von TCP verwendet
- Emails werden über Mail Transfer Agents (MTA) ausgetauscht
- jeweilige Anwendungsprogramme (Netscape Messenger / Outlook / sendmail) übernehmen Kommunikation mit den MTA´s
- Die MTAs verwenden zur Kommunikation untereinander einfache ASCII-Zeichen. Der Client sendet Kommandos zum Server, der mit einem numerischen Code und einem optionalen String antwortet.

Nachteil von SMTP:

- keine weiteren Informationen über den Verbleib der Email
- bei Fehlern werden meist nur nichtstandardisierte Meldungen zurück geschickt

Quelle(n) : www.tecchannel.de

SMTP – Kommandos

- HELO oder HELLO: Startet eine Sitzung und identifiziert den Client am Server
- MAIL Startet eine Mailübertragung
- RCPT Recipient: Identifiziert den Empfänger (forward-path) Bei mehreren Empfängern wird das Kommando mehrmals ausgeführt.
- DATA Der Server antwortet auf das Kommando mit dem Code 354 und wartet auf die Übertragung der Nachricht. Der Client beendet die Übertragung mit "CRLF"."CRLF".
- RSET Reset: Abbruch der Mailtransaktion wird abgebrochen. Die Verbindung zwischen beiden Rechnern bleibt jedoch bestehen.
- VRFY Verify: Überprüft eine Empfänger-Adresse.
- NOOP dient zum Halten der Verbindung (Server sendet "250 OK")
- QUIT Beendet die Verbindung. Der Server muss daraufhin die Antwort "250 OK" zurückliefern.

Quelle(n) : www.tecchannel.de

SMTP - Antwortcodes

- mit den SMTP: Antwortcodes beantwortet der Email-Server jede Anfrage vom Email-Client
- Die wichtigsten Codes (nicht vollständig)
  - 211 System-Status oder System-Hilfe.
  - 214 Hilfe - Informationen zum Ausführen eines Kommandos.
  - **220 Server bereit.**
  - 221 Server beendet Verbindung.
  - **250 Kommando ausgeführt.**
  - 251 Keine lokale Mailbox; Weiterleitung an "forward-path".
  - 354 Starte Empfang der Mail; Beenden mit "CRLF". "CRLF".
  - 421 Service nicht verfügbar; Verbindung wird beendet.
  - 450 Aktion nicht ausgeführt - Mailbox nicht verfügbar.
  - **500 Syntax-Fehler - Kommando unbekannt.**
  - 501 Syntax-Fehler - Parameter oder Argument falsch.
  - 502 Kommando unbekannt / nicht implementiert.
  - 503 Falsche Reihenfolge der Kommandos.
  - 504 Parameter unbekannt / nicht implementiert.

Quelle(n) : www.tecchannel.de

Aufbau einer Email

Eine E-Mail besteht aus drei Teilen:

- **Envelope** (Umschlag) mit Sender und Empfänger der Nachricht und wird von den Mail Transfer Agents benötigt.
- **Header:** Verwendet der Mail-Client für weitere Informationen wie Client-Kennung und Message-ID.
- **Body:** Enthält den eigentlichen Text der Nachricht. RFC822 spezifiziert den Body als ASCII-Text.
- Beim Versenden einer Mail mit dem Kommando DATA überträgt der Client den Header, gefolgt von einer Leerzeile und dem Body.
- **Jede übertragende Zeile darf nicht länger als 1000 Bytes sein.**

Aufbau einer Email - Headers

- Beispiel :

> Received: by xyz.de. id AA00502; Mon, 19 Nov 2001 12:47:32 +0100
> Received: from adam1 (715684625313-0001@[192.168.80.201]) by fwd00.xyz.de
> with smtp id 166Cyz-1KXYRsC; Tue, 20 Nov 2001 16:38:45 +0100
> From: adam@xyz.de (Adam)
> To: eva@test.de (Eva)
> Subject: Beispiel-Mail
> Date: Mon, 19 Nov 2001 12:47:31 +0100
> Reply-To: adam@xyz.de
> Message-ID: <9307191947AA00502.Adam@xyz.de>
> MIME-Version: 1.0
> Content-Type: text/plain; charset="iso-8859-1"
> Content-Transfer-Encoding: 8bit
> X-Mailer: Microsoft Outlook IMO, Build 9.0.2416 (9.0.2910.0)

- Unter Received werden alle SMTP-Server eingetragen, die die E-Mail auf dem Weg vom Sender zum Empfänger passiert hat
- jede Email hat eine eindeutige Message-ID bestehend aus Nr. und Hostnamen
- mit X-beginnende Zeilen sind in der Regel vom Mail-Client hinzugefügt und für den Versand der Nachricht nicht zwingend erforderlich

Quelle(n) : www.techchannel.de

SMTP – Beispielablauf beim Versenden einer Email

Kommunikation (C=Clienteingabe S=Server)	*Bemerkung*
> S: 220 test.de SMTP server ready	*Server meldet sich bereit*
> C: HELO xyz.de.	*Client stellt sich vor*
> S: 250 xyz.de., pleased to meet you	*Server akzeptiert*
> C: MAIL From:adam@xyz.de	*Client sendet Empfänger*
> S: 250 <adam@xyz.de> Sender ok	*Server akzeptiert*
> C: RCPT To:<eva@test.de>	
> S: 250 <eva@test.de> Recipient ok	
> C: RCPT TO:<tom@test.de>	
> S: 250 <tom@test.de> Recipient ok	
> C: DATA	*Client startet Emailtext*
> S: 354 Enter mail	
> C: Hallo Eva, hallo Tom!	
> C: Beispiel für den Mail-Versand mit SMTP.	
> C: Adam	
> C: .	*Ende CRLF . CRLF*
> S: 250 Mail accepted	
> C: QUIT	
> S: 221 test.de delivering mail	

Quelle(n) : www.techchannel.de

SMTP – MIME - Multipurpose Internet Mail Extension

Historisch entstandenes Probleme der Kodierung des Emailinhaltes

- es werden im Email-Body nur 7-Bit- ASCII-Textzeichen akzeptiert
- keine Kodierung internationaler Sonderzeichen (wie z.B. deutsche Umlaute äüöß) möglich

Ausweg mit RFC2045 zur Def. von Multipurpose Internet Mail Extension)

- Body wird weiterhin als ASCII-Text mit einem zusätzlichen MIME-Header übertragen

MIME-Header :

MIME-Version 1.0 - kennzeichnet MIME-Version.

Content-Type: text/plain; charset=iso-8859-1

- Bestimmt den Inhalt der Mail (siehe Folgeseite) – bei Typ "text" und "multipart" wird eine Zeichensatzangabe und Textkörper-Kennung ergänzt

Content-Transfer-Encoding: 8bit

- Kennzeichnet den Algorithmus, in dem die Daten vorliegen, z.B. 7bit, 8bit, binary, quoted-printable

Quelle(n) : www.techchannel.de

SMTP – MIME-Typen

- Hinweis: Diese Inhaltsklassifizierungen wurden später auch beim HTTP-Protokoll angewendet und erweitert.
- MIME stellt auch gleichzeitig die Informationen zur Verfügung, die ein E-Mail-Client zur Auswahl des Darstellungsprogramms benötigt.
- MIME unterteilt die Datentypen (Media-Types) in sieben Obergruppen mit mehreren Untergruppen. Jede Dateiendung wird einem "Media-Type" zugewiesen.

Wichtige MIME-Typen (nicht vollständig)

- text plain - unformatierter Text.
- richtext - Text mit einfachen Formatierungen, zum Beispiel Kursiv.
- enriched - Vereinfachte und erweiterte Form von richtext.
- multipart mixed - Mehrere Body-Teile, die sequenziell bearbeitet werden.
- digest - Auszug aus einer E-Mail.
- external-body - Inhalt ist ein Zeiger zur eigentlichen Nachricht.
- application octet-stream - Binär-Daten.
- postscript - PostScript-Daten.
- image jpeg ISO10918-Format - Grafikdatei
- gif CompuServe Graphic Interchange Format (GIF).
- audio basic Kodiertes 8-Bit-µ-law Format.
- video mpeg ISO11172-Format

Quelle(n) : www.techchannel.de

Empfangen von Mails mit POP3 oder IMAP

Problem beim Empfangen von Emails:

- Anwender müsste eigenen SMTP-Email-Server verwalten
- bei Anbindung über Wählverbindung nicht möglich, da nicht ständig online

Lösung : Nachrichten werden beim Provider zwischengespeichert

- zwei Protokolle Für den Fernzugriff auf die E-Mails
  - erste Version POP (seit 1984), aktuell POP3
  - zweite Ansatz ist das 1986 entwickelte IMAP Version 4,

Optionen für verteilten E-Mail-Strukturen nach RFC1733:

Offline-Verarbeitung: Mails an einen Server übermittelt, Client lädt die Nachrichten herunter und bearbeitet die Post lokal auf dem Rechner

bei **Online-Betrieb** bleibt die Mail auf dem Server und wird dort vom Client bearbeitet.

Disconnected-Zugriff, ein Hybrid aus Online- und Offline-Modell. Bei dieser Variante nimmt der Client Verbindung zum Server auf, erstellt eine Kopie der Nachrichten und baut die Verbindung wieder ab. Nachdem der Benutzer die Mail bearbeitet hat, erfolgt ein Abgleich der Mails zwischen Client und Server.

Quelle(n) : www.techchannel.de

POP und IMAP im Vergleich

IMAP	POP3	Funktion
X	X	Unterstützung für Offline-Modus
X		Unterstützung für den Online-Modus
X		Unterstützung für den Disconnected-Modus
X	X	Mails laufen beim SMTP-Server auf
X	X	für den Empfang von Nachrichten ist kein SMTP-Gateway erforderlich, jedoch für den Versand
X		Zugang zu unterschiedlichen Mailboxen möglich
X		Erlaubt Zugang zu anderen Daten, wie beispielsweise News
X		Hohe Performance bei Verbindungen über Leitungen mit niedriger Bandbreite (Modems)
X		Message-Status-Flags lassen sich bearbeiten
X	X	Nachrichten sind mit unterschiedlichen Clients im Netz zugänglich
X	X	Offene Protokolle; Spezifiziert in RFCs
X		Zusatzprotokoll für die Verwaltung von Benutzereinstellungen

Fazit: IMAP bietet im Vergleich zu POP zahlreiche Vorteile, besonders für den Fernzugriff auf E-Mails.

Quelle(n) : www.techchannel.de

Eine POP3-Beispielsitzung

> S: +OK POP3 server ready	*Server ist bereit*
> C: user tecchannel	*User meldet sich an*
> S: +OK	
> C: pass ahd635d	*User sendet Passwort*
> S: +OK	
> C: LIST	*User fragt Status ab*
> S: +OK 2 messages (320 octets)	
> S: 1 120	
> S: 2 200	
> S: .	
> C: RETR 1	*User fordert Email 1 an*
> S: +OK 120 octets	
> S: <Server sendet Nachricht 1>	
> S: .	
> C: DELE 1	*User verlangt Löschung*
> S: +OK message 1 deleted	
> C: DELE 2	
> S: +OK message 2 deleted	
> C: RETR 3	
> S: -ERR no such message	
> C: QUIT	
> S: +OK	*Server führt jetzt Aktionen aus !*

Quelle(n) : www.techchannel.de

Probleme mit Massen-Emailversand (SPAM)

- Spammer nutzen häufig fremde SMTP-Server zum Versenden der Mails (Relaying) :
- in vielen Ländern illegal und Serverbetreiber kommt dadurch in Bedrängnis: , sondern bringt unnötige Datenvolumen zu bezahlen und Störung der Infrastruktur
- bei häufigem unautorisiertem Relaying wird der gesamte Server auf Blacklist gesetzt wird. andere MTAs akzeptieren von diesem Server dann KEINE Mails mehr

Anti-Spam-Maßnahmen : Verhindern von Relaying

- SMTP-Server soll unautorisiertes Relaying erkennen und ablehnen

Maßnahmen durch Identifizierung von Sender und Empfänger einer Mail

- HELO Hostname - es kann jeder beliebige Hostname angegeben werden – UNSICHER !
- MAIL From: Der Client kann jede beliebige Adresse angeben – UNSICHER !
- RCPT To: dies muss eine korrekte Adresse sein - Überprüfbar , d.h. Adressaten innerhalb der eigenen Domäne werden akzeptiert !
- SMTP_CALLER IP-Adresse des Client - Überprüfbar

Bei vielen Internetprovidern wird eine Kombination mit POP3 durchgeführt:

- Beim Abrufen von Nachrichten per POP3 oder IMAP4 muß Client den Benutzernamen und Passwort angeben und überträgt auch seine IP-Adresse
- dieser IP-Adresse wird für bestimmte Zeit ein Versand über SMTP erlaubt !
- Das Problem der Übertragung der Passwörter im Klartext kann durch eine sichere Übertragung mit dem TLS (Transport Layer Security) nach RFC2487 behoben werden.

Quelle(n) : www.techchannel.de

FTP - Das File Transfer Protokoll

FTP wurde bereits ab 1971 entwickelt (erste RFC 114)

- heute gültige Spezifikation RFC 959 wurde bereits 1985 veröffentlicht
- dient fast ausschließlich zur Übertragung von Dateien

Grundprinzip

- FTP erzeugt zwischen Server und Client zwei Verbindungen :
  - die erste dient zur Steuerung und überträgt nur FTP-Kommandos und entsprechende Antworten in der Regel auf TCP-Port 21
  - die zweite Verbindung überträgt die Daten in der Regel auf TCP-Port 20
- FTP-Client und FTP-Server tauschen Text-Basierte Kommandos aus
- Steuerung erfolgt durch User Protocol Interpreter
  - dieser startet die Verbindung
  - sendet seine Befehle an den Server
  - und bearbeitet dessen Antworten.

Bei Start der Datenübertragung erfolgt die Steuerung des Transfers durch den jeweiligen Sender (Server bei Download, Client bei Upload)

Ablauf einer FTP-Verbindung

Aufbau der Verbindung und Authentifizierung

- Client baut die Verbindung zum FTP-Server auf. Alle FTP-Implementierungen müssen dabei die Standard-Ports 20 und 21 unterstützen. Verbindungen zu Nicht-Standard-Ports darf nur die Client-Seite initiieren. Optionen :
  - **Aktives FTP** - Client öffnet einen zufälligen Port (typisch >1023) und teilt dem Server diesen und die eigene IP-Adresse mit. Die Datenübertragung auf der Server-Seite erfolgt dabei über Port 20. Befehlskommunikation erfolgt nur auf dem Control Port, auch parallel zur Datenübertragung.
  - **Passives FTP** - Der Client sendet ein PASV-Kommando, der Server öffnet einen Port und übermittelt diesen mitsamt IP-Adresse an den Client. Passives FTP ist notwendig, wenn der Server selbst keine Verbindung zum Client aufbauen kann (z.B. wegen Firewall über Ports oder Router mit NAT)
- Bei Erfolg gibt FTP-Server Informationen über das System aus "Welcome at ..." und fragt Username und Passwort ab.
- Viele öffentliche FTP-Server unterstützen sogenanntes anonymes, frei zugängliches FTP mit dem Username "anonymous" und beliebiges Passwort (Vorsicht heute !!)
- nach erfolgreicher Anmeldung erfolgt Aufbau einer Kontrollverbindung
- beim Übertragen von Daten wird ein Datenkanal geöffnet
- zulässige Aktionen sind abhängig von den Rechten des Benutzers
- Trennung der Verbindung erfolgt explizit durch Client oder Server (Timeout)

Datenübertragungen mit FTP

- Bei der Datenübertragung wird auf TCP gesetzt und auf dessen Fehlerkontrolle vertraut. Es findet **keine Fehlerkontrolle durch FTP** statt
- im FTP-Protokoll existiert jedoch ein Verfahren zur Wiederaufnahme von unterbrochenen Übertragungen (Ausfall der Verbindung durch Überlastung oder technischen Problemen) – das **Resuming**
- **meist nur unterstützt durch neuere FTP-Server und –Clients**
  - in den einzelnen Datenpaketheadern sind Restart-Markierungen
  - Kommt es zum Abbruch und einer erneuten Verbindungsaufnahme, so sendet der Client ein Restart-Kommando, der FTP-Server antwortet mit einem Restart Marker Reply
  - der Server übermittelt die zuletzt gesetzte Markierung
  - Server und Client gleichen die Markierungen über das MARK-Kommando ab
  - der Client startet dann ab der vereinbarten Markierung erneut
- **FXP - Zusatzoption** zum Abgleich von FTP-Servern mit externer Steuerung
  - Das File Exchange Protocol (FXP) erlaubt eine durch den Client gesteuerte Dateiübertragung zwischen zwei Servern, was Zeit spart.
  - FXP verwendet Active Mode und Passive Mode gleichzeitig: Client sendet an 1. Server ein PASV-Kommando und erhält dessen IP und Port. Diese sendet der Client an den zweiten Server, wodurch dieser eine Verbindung zum ersten Server aufbaut. Der Client kann von 3. Seite die Übertragung steuern.

Datenübertragungen mit FTP – Formate

Übertragungsarten

- Streammodus - überträgt die Daten als eine Einheit vom FTP-Server zum Client beziehungsweise in umgekehrter Richtung.
- Blockmodus - FTP-Server zerlegt die Dateien in Datenblöcke
  - Jeder Datenblock ist mit einem Header mit Informationen zur Länge und Markierungen zum Resuming versehen.
  - nach jedem Block kommt eine End-of-Record-Markierung (EOR)
  - nach der vollständiger Übertragung folgt End-of-File-Markierung (EOF)
- Compressed-Modus überträgt Daten in komprimierter Form mit dem Ziel einer Reduzierung der zu übertragenden Datenmengen
  - Kompression ist relativ einfach: Run-Length-Kodierung mit Zusammenfassung , direkt aufeinander folgender identischer Bytes zu zwei Bytes (stärkere Kompressionen wie LZW-Verfahren ist nicht vorgesehen)
  - Besser also erst Zippen und dann per FTP übertragen !

Client und Server einigen sich vor Übertragung auf Kodierungsart:

- ASCII-Modus dient zur Übertragung von Textdateien (ist Standardmodus!)
- EBCDIC-Code (veraltete Textcodierung bei Großrechnern)
- Binärmodus (Binary) ist für Nicht-Textdateien vorgesehen

Die wichtigsten FTP-Befehle

Befehle für die Zugriffskontrolle

> USER Username - zur Angabe des Usernamens

> PASS Passwort - zur Angabe des Passworts (Achtung: im Standardmodus wird Passwort nicht verschlüsselt !!)

> ACCT Account- zusätzliche Angabe eines Account-Namen, für spezielle Aktionen, wie Speichern oder Löschen von Daten

> AUTH (Authentication/Security Mechanism)

> ADAT (Authentication/Security Data)
Über diesen Befehle tauschen Client und Server für einen Sicherheitsmechanismus notwendige Daten aus.

- PROT (Data Channel Protection Level)
Dieser Befehl zeigt die Art des Datenkanalschutzes an.

- > REIN (Reinitialize)
Dieses Kommando hebt die Berechtigung des Users auf. Nach dem Befehl ist ein erneuter Login mit USER und PASS notwendig. Bereits gestartete Datentransfers sind von dem Befehl nicht betroffen.

> QUIT (Logout) - Beendet die Verbindung zwischen Client und Server

Quelle(n) : www.techchannel.de

Die wichtigsten FTP-Befehle 2

Servicebefehle

> RETR (Retrieve) – Überträgt eine Datei von Server zum Client.

> STOR (Store)– speichert eine Datei auf (Achtung: Überschreibt vorhandene Dateien)

> CWD (Change Working Directory) – geht auf Server in anderes Verzeichnis (analog zum DOS/Unix CD-Befehl.)

> DELE (Delete) – Löscht eine Datei auf dem Server.

> RMD (Remove Directory) – Löscht ein Verzeichnis auf dem Server !!

> MKD (Make Directory) – erzeugt Serverseitig ein Verzeichnis.

> PWD (Print Working Directory) – Anzeige aktuelles Verzeichnis

> LIST – listet Dateien im aktuellen Verzeichnis des Servers auf

> NLST (Name List) – Sendet eine Verzeichnis-Listing an den Client.

> SYST (System)– zeigt Betriebssystem des FTP-Servers an

> STAT (Status) – Übermittelt den Status eines Systems.

> HELP oder ? – gibt die Hilfetexte des Servers aus.

Quelle(n) : www.techchannel.de

Eine Beispiel FTP-Session

```
ftp> open localhost
Verbindung mit mobil2.informatik.htw-dresden.de wurde hergestellt.
220 Personal FTP Professional Server ready
Benutzer (mobil2.informatik.htw-dresden.de:(none)): wiedem
---> USER wiedem
331 Password required for wiedem.
Kennwort:
---> PASS XXXXXX   (unverschlüsselte Übertragung bei FTP)
230 User wiedem logged in.
ftp> pwd
---> XPWD
257 "c:/htdocs/" is current directory.
ftp> cd xml
---> CWD xml
250 CWD command successful. "c:/htdocs/xml/" is current directory.
ftp> binary
---> TYPE I
200 Type set to I.
ftp>
```

Sicherheit bei FTP

- Die FTP-Spezifikation sieht bis auf die einfache Benutzer-Authentifizierung, die **Kennung und Passwort unverschlüsselt** zwischen Client und Server übermittelt, keinerlei Sicherheitsfunktionen vor.
- deshalb sicherheitsspezifische Erweiterungen mit RFC 2228 (FTP Security Extensions) für die Benutzer-Authentifizierung, Integrität, Vertraulichkeit der Daten- und Kontrollkanäle durch Verschlüsselung von Befehlen, Replys und der Daten.

Schwerpunkt ist Benutzer-Authentifizierung :

- Client teilt Server gewünschten Sicherheitsmechanismus mit AUTH-Befehl mit
- Server akzeptiert diesen Mechanismus oder aber lehnt ihn ab
- Client kann mehrere Mechanismen testen, um eine von beiden Seiten unterstützte Methode zu finden
- Protokollabhängige Authentifizierungsinformationen werden mit ADAT-Befehl (teilweise auch im Handshake) übertragen
- Nach Einigung auf gemeinsames Verschlüsselungssystem, werden sichere (verschlüsselte) Übertragungskanäle angelegt

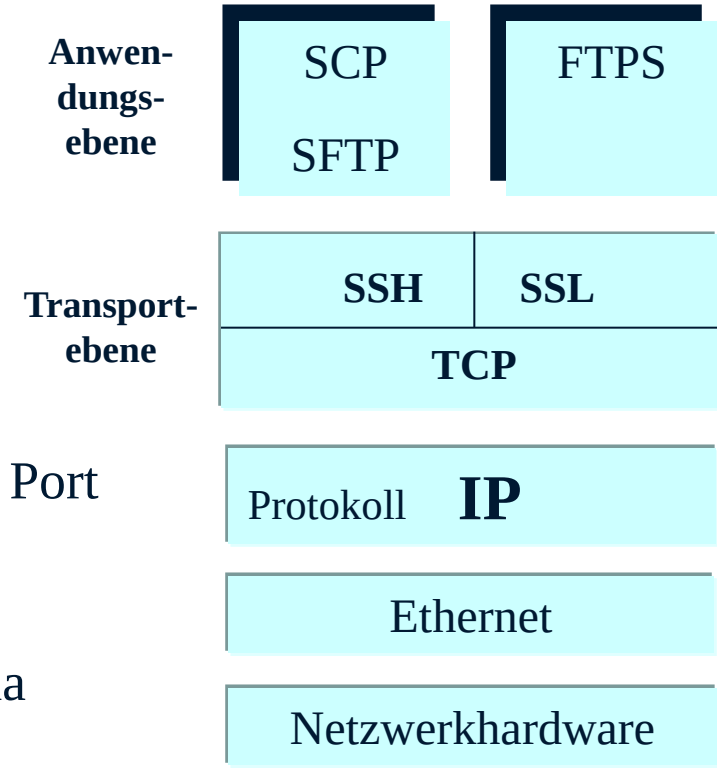
Quelle(n) : www.techchannel.de

Sichere Dateiübertragung

- **Standard-FTP ist unsicher (Auth. & Daten unverschlüsselt) !**

Lösungsoptionen :

- **Secure Shell (SSH)** - Erstellung einer verschlüsselten Verbindung über unsichere Netzwerkkanäle (als Ersatz für ältere Telnet-Protokolle) - Version SSH-1 von 1995 mit Schwachstellen, SSH-2 stark überarbeitet und sicher, OpenSSH als Opensource-Version
- **SCP - Secure Copy Protocol** sichert Auth. und Datenübertragung über einen SSH-Kanal (Secure Shell) -> Client WinSCP
- **SSH File Transfer Protocol** (**SFTP**) – Weiterentwicklung von SCP
- **Secure File Transfer Protocol - (Secure FTP)** sichert nur die Authentifizierung, der Datentransfer läuft unverschlüsselt über einen zufällig definierten Port
- **FTP über SSL** (**FTPS**) sichert die gesamte Kommunikation über eine SSL-Verschlüsselung (meist AES 256 bits) ab, Client /Server -> Filezilla



Das SSL- und TLS– Protokoll

Historie und aktueller Stand :

- SSL (Secure Sockets Layer) wurde 1994 von Netscape entwickelt
- erster Einsatz im Netscape Navigator 3.0
- 1999 wurde auf Basis von SSL der neue Standard **Transport Layer Security TLS** entwickelt (= V3.1 von SSL, TLS ist abwärtskomp. zu SSL (SSL2.0 und 3.0. sind seit 2015 veraltet definiert!) -> **aktuell TLS 1.3**

Funktionsweise von TLS:

- Über das **TLS Handshake Protocol** wird die Art der Verschlüsselung ausgehandelt (Zwei-Wege-Authentifizierung mittels Zertifikaten / Identifikation und Authentifizierung auf Basis asymmetrischer Verschlüsselungsverfahren und Public-Key-Kryptografie)
- Ergebnis eines erfolgreichen TLS-Handshakes ist ein EINMAL-Schlüssel für die jeweilige Session (mit Zufallszahlen und Prüfsummen abgesichert)
- Im darunter liegenden **TLS-Record Protocol** wird dann mir dem Einmal-Schlüssel eine symmetrische (schnellere) Verschlüsselung mit DES, Triple DES oder AES realisiert.
- optional können die Daten fragmentiert, komprimiert und mit Prüfsummen versehen werden.

Das Hypertext Transfer Protocol

Historie :

- der erste HTTP- Server wurde 1989 am CERN Genf, dem europäischen Kernforschungsinstitut von Tim Berners-Lee entwickelt
- Anlaß und Motivation war bessere Bereitstellung von Forschungsergebnissen mit Schwerpunkt auf grafischen Ergebnissen
- Tim Berners-Lee entwickelte sowohl das HTTP-Protokoll zur Übertragung der Informationen wie auch HTML zur Kodierung der Daten

Grundprinzipien von http

- Relativ einfaches Protokoll und auch einfach zu implementieren

Übermittlung der Daten erfolgt nach dem Request-Response-Schema

- Alle Daten werden ähnlich wie bei Emails als ASCII-Text übertragen
- Ein HTTP-Client (z.B. Webbrower) sendet seine Anfrage (Request) an den HTTP-Server.
- HTTP-Server nehmen standardmäßig auf TCP-Port 80 Anfragen entgegen
- Eine Anfrage besteht immer aus einem Request-Header und optionalen Header, optional gefolgt von Daten (z.B. Aus Formularfeldern)
- Der Server dekodiert die Anfrage, holt die Daten oder ruft ggf. andere Hilfsprogramme zur Verarbeitung oder Datenbankabfrage auf.
- Die Antwort (Response) wird als ASCII-Text kodiert und an den Client zurückgeschickt.
- Die Antwort besteht ebenfalls wieder aus Response-Header und Daten.
- Im Gegensatz zu FTP sieht HTTP beim Verbindungsaufbau keine mehrstufige Handshake-Phase vor, es wird sofort ein Request gesendet und danach auf die Antwort gewartet.
  - Vorteil: sehr einfach und schnell
  - Nachteil: keine dauerhafte Verbindung zwischen Client und Server und damit Probleme mit Sitzungsorientierten Anwendungen

HTTP-Versionen

HTTP 0.9 (nicht mehr relevant)

- erstes und sehr einfaches Transportprotokoll für Daten (nur GET-Methode)

HTTP 1.0 (RFC 1945)

- neues Request-Response-Format mit mehr Optionen
- nach dem Request folgt ein Satz von Headerfeldern, z.B. zur Übermittlung des Browsertyps
- Unterstützung der POST-Methode zur Übertragung von Daten und HTML-Formulareingaben vom Browser zum Webserver

HTTP 1.1 (RFC 2616)

- ist abwärtskompatibel zu HTTP 1.0, d.h. unterstützt alle HTTP-Funktionen
- auch alle http 1.1-Clients (Browser) unterstützen HTTP 1.0

HTTP 2.0 (RFC 7540 und 7541) seit 2015

- ist abwärtskompatibel zu HTTP 1.1

Wichtige Verbesserungen:

- **weitere Optionen zum Zusammenfassen von Anfragen (wie schon bei 11.1)**
- **Zusätzliche und Datenkompressionsmöglichkeiten auch binär kodierte Übertragung von Inhalten**
- **Server-initiierte Datenübertragungen (push-Verfahren), speziell für mobile Push-Nachrichten (auch zum Reduzieren der Connects …)**

HTTP-Requests

Ein Request besteht aus Header und optionalen Teilen mit folgende Struktur :
> METHOD URL HTTP/version
> General Header
> Request Headers
> Entity Header (optional)
> Leerzeile
> Request Entity (falls vorhanden)
>

Die Methode gibt den grundlegenden Typ der Anfrage an (siehe HTTP-Methoden)

Beispiel zum einfachen Abruf einer Seite :
> GET http://www.bla.de/verzeichnis1/seite2.html HTTP/1.1
> Date Thursday, 14-Oct-16 17:55 GMT
> User-agent: Mozilla/4.6
> Accept: text/html, text/plain
>

Quelle(n) : www.techchannel.de

HTTP - Methoden

- Die HTTP-Methode bestimmt den grundlegenden Typ der Anfrage.

HTTP 1.1. definiert 8 acht Methoden :

1. GET- Methode (am häufigsten verwendet)

- zur **Anforderung eines Dokuments** oder einer anderen Quelle
- eine Quelle wird durch den Request-URL identifiziert
- zwei zusätzliche GET-Untertypen:
- conditional GET zur Anforderung von Daten mit Bedingungen
  - Bedingungen im Header-Feld "Conditional" , z.B. Bedingung If-Modified-Since, If-Unmodified-Since oder If-Match
  - deutliche Verringerung der Netzbelastung, da nur geänderte Daten übertragen werden und ungeänderte Daten aus dem Cache des Browsers oder Prxies geholt werden
- partielle GET-Methode zur teilweisen Übertragung von Daten (hauptsächlich zur Wiederaufnahme eines unterbrochenen Datentransfers)

2. POST-Methode

- **für Datenübertragungen zum Server**
- Daten aus Formulareingaben, Uploads von Dateien, Kommentierung bestehender Quellen, Übermittlung von Nachrichten an Foren
- Ablage der Daten in der Entity-Sektion enthalten
- Die URL der POST-Methode dient zur Auswahl der Verarbeitungsroutine

HTTP – Methoden II

3. OPTIONS-Methode

- zum Abruf von verfügbare Kommunikationsoptionen, erlaubt Test auf mögliche Beschränkungen durch Server oder Proxy ohne direkte Datenübertragung

4. HEAD-Methode

- zum alleinigen Abruf der Headerinformationen (wie GET ohne Daten)

5. PUT-Methode (selten, meist wird dafür POST verwendet)

- zur Modifikation bestehender Quellen oder Erzeugung neuer Daten auf dem Server

6. DELETE-Methode (selten verwendet)

Zum Löschen von Daten auf dem HTTP-Server gelöscht

7. **TRACE-Methode**

- Zum Debuggen von Requests bei mehreren Servern in größeren Serverfarmen
- Max-Forwards-Header-Feld bestimmt maximale Anzahl an Hops
- Header-Feld "Via" protokolliert alle durchlaufenen Server

8. CONNECT-Methode

- Zum Aufbau von Tunnel-Verbindungen über Proxyserver
- Hauptsächlich zum Aufbau verschlüsselte Verbindungen
- Details zu diesem Verfahrenim Internet Draft " Tunneling TCP based protocols through Web proxy servers"

HTTP- Response –Codes

- Antwort eines HTTP-Servers besteht aus Statuszeile und Response-Header-Feldern
- Statuszeile gibt Protokollversion, Status Code und Reasons Phrase an
  - Status Code ist dreistelliger Integer-Wert aus fünf Kategorien :

    1xx: Informelle Meldungen: Request erhalten, Bearbeitung wird durchgeführt.

    2xx: Erfolg: Request wurde erfolgreich erhalten, verstanden und angenommen.

    3xx: Weiterleiten: es ist eine Weiterleitung notwendig

    4xx: Clientfehler: Request enthält ungültige Syntax oder ist nicht bearbeitbar

    5xx: Serverfehler: Der Server kann eine gültige Request nicht bearbeiten.

  - Die Reasons Phrase enthält die Klartext-Bezeichnung der Meldung.

Besonders häufige und damit sehr bekannte Meldungen sind:

404 für "File not Found" (Datei nicht gefunden)

403 für "Forbidden" (Zugriff verweigert)

HTTP-Response- Beispiel

Der Aufbau einer HTTP-Response ist ähnlich zur Request:

```
> HTTP/version Status-Code Reason-Zeile
   > General Header
   > Response Header
   > Entity Header (optional)
   > ______Leerzeile_______
   > Resource Entity (falls vorhanden)
   >
```

Beispiel :

```
   > HTTP/1.1 200 OK
   > Via: HTTP/1.1 proxy_server_name
   > Server: Apache/1.3
   > Content-type: text/html, text, plain
   > Content-length: 78
   >
   > <html>
   > <head>
   > <title>HTTP</TITLE>
   > </head>
   > <body>
   > <p> HTTP/1.1-Demo</p>
   > </body>
   > </html>
   >
```

Statuszeile mit OK-Status

MIME-Type

Datenlänge

Leerzeile !

Start der Daten im HTML-Format

Quelle(n) : www.techchannel.de

Authentifizierung über HTTP

- HTTP-Spezifikation definiert zwei ähnliche Authentifizierungsmethoden des Anwenders HTTP-Server (Basic HTTP Authentication) und bei zwischengeschalteten Proxyserver (Proxy Server Authentication)
- Funktionsweise ist sehr ähnlich, : Unterschiede gibt es bei den verwendeten Status Codes und den Headernamen

Basic HTTP Authentication

- verwendet Benutzerkennung und Passwort für die Authentifizierung
- Erster Request wird OHNE Authentifizierungsinformationen gesendet
- Server antwortet mit spezieller Statusmeldung und fordert mit WWW-Authenticate-Header-Feld eine Angabe der Zugangsdaten
- Authenticate – Realm definiert den Absender der Authentifizierungs-anforderungen (meist der Name des Bereiches oder der Anwendung)
- zur Kodierung von Kennung und Passwort dient Base64 –Verfahren – eine nur sehr einfache Bitverschiebung !!! **-> unsicher**
- Bsp.: Zweiter GET-Befehl mit Angabe des Passwortes

```
> GET /verzeichnis_x/seite_y.html HTTP/1.1
> User-agent: Mozilla/4.6
> Accept: text/html, image/gif, image/jpeg
> Authorization: Basic KDWkfowsOm=
>
```

Alle weiteren Requests werden immer mit dieser Kennung gesendet !

Sichere Kommunikation über HTTPS

- Die normale HTTP-Kommunikation ist unverschlüsselt und damit unsicher.

Lösungsoption : HTTPS

- Verschlüsselung analog zu FTPS mittels SSL/TLS
- meist Kommunikation über Port 443 statt 80
- Das HTTPS-Protokoll selbst ist identisch zu HTTP !

Probleme mit HTTPS

- Die SSL-Verschlüsselung ist relativ rechenintensiv und sollte daher nicht standardmäßig für alle, sondern nur für sicherheitssensitive Bereiche verwendet werden (Shopkasse / Homebanking)
- **allerdings werten Google & Co. seit 2017 Webseiten ohne HTTPS als unsicher und werten diese im Ranking ab !**
- pro Hostname ist GENAU eine IP-Adresse erforderlich, damit ist eine Einrichtung von virtuellen HTTP-Servern (Angabe des Hostnamens im HTTP-Header -> mehrere Hostnamen auf einer IP-Adresse -> Massenprovider) nicht möglich.

Weitere Anwendungsprotokolle

Neben den betrachteten Protokollen existieren noch weitere Protokolle :

- News – Protokoll zum Abgleich von News-Servern
- veraltet und wahrscheinlich kurz vor dem Aussterben:
  - Archie, Finger, Whois Hilfsdienste zur Suche nach Dateien, Personen und Adressen
  - Wais verteilte Informationsdatenbank mit einheitlicher Abfrage
  - Gopher universelles, text-basiertes Informationssystem

- Das Grundprinzip entspricht jedoch immer den bereits betrachteten Protokollen