Einführung

Was ist das Internet?

Komponentensicht
1. *Computer* – führen Netzwerkanwendungen aus
2. *Kommunikationsmedien* – Kupferkabel, Glasfaser, Funk
3. *Zwischensysteme* – Weiterleitung durch Router und Switches

Dienstsicht
⇒ Infrastruktur, die Dienste bereitstellt
- *Kommunikation* (Mail, Messaging, soziale Medien)
- *Information* (Surfen)
- *Unterhaltung* (Streaming, Spiele)

Rand des Internet

Geräte
- Clients
- Server

Zugangsnetze
- Heimnetz
- Mobiles Zugangsnetz
- Unternehmensnetz

Kern des Internet

Pakete: voneinander unabhängige Einheiten für die Weiterleitung – werden durch Netz zur Zielanwendung geleitet

Anwendungsschicht

Historie

70er/80er:
- textbasierte Anwendungen

90er:
- World Wide Web
- Instant Messaging
- P2P-Filesharing

seit 2000: steigende Vielfalt + Allgegenwärtigkeit
- Streaming (Spotify, YouTube)
- Gaming
- Soziale Netzwerke
- Smartphones

Grundlagen — Schichtenmodell

Kommunikation in Schichten organisiert
Anwendungsschicht: oberste Schicht
- enthält Anwendungsprotokolle
- Anwendung kümmert sich nicht um Datentransport

Datentransport: unter Anwendungsschicht liegende Schichten
- Interna für Anwendung transparent
- Verzögerungen bleiben vor Anwendung verborgen

Grundlagen — Verzögerung

Abhängig von
- *Ausbreitungsverzögerung* t_a
- *Sendezeit* t_s
- *Pufferfüllstände*

Ausbreitungsverzögerung $t_a = \frac{d}{v}$
- *Zeitspanne zwischen Absenden eines Signals und dessen Eintreffen am anderen Ende des Mediums*
- Abhängig von:
  Ausbreitungsgeschwindigkeit v
  Länge des Mediums d

Sendezeit $t_s = \frac{X}{r}$
- *Zeit zwischen Beginn und Abschluss der Sendung*
- Abhängig von:
  Datenmenge X
  Datenrate des Mediums r
- **Achtung**: Nach Sendungsabschluss sind die Daten noch nicht beim Empfänger!
  ⇝ Ausbreitungsverzögerung t_a

Verzögerung im Router
- *Pufferung* der Daten in Warteschlange
- *Verarbeitung* (Fehlerüberprüfung usw.)

Grundlagen — Protokollstack

Application: SMTP, HTTP, XMPP,…
Transport: TCP, UDP
Network: IP
Data Link: Ethernet, 802.11 (WiFi)
Physical: Bits auf Medium

Grundlagen — Prozess und Nachricht

Prozess: Programm, das im Endsystem (Anwendungsschicht) abläuft
Nachricht: Ausgetauscht zwischen Prozessen auf *unterschiedlichen* Endsystemen

Grundlagen — Socket und Interface

Programmierschnittstelle für verteilte Anwendungen
Von OS bereitgestellte API
Anwendungsprozess sendet/empfängt Nachrichten zum/vom Socket
Portnummern: (De-) Multiplexing auf Endsystemen
- viele Prozesse auf Endsystem kommunizieren gleichzeitig über Netzwerk
  ⇝ eindeutige Socket-Identifikation über Portnummer

Grundlagen — Client-Server-Anwendungen

Server:
- ständig in Betrieb
- permanente IP-Adresse
- häufig in Datenzentren

Clients:
- kommunizieren mit Server
- kommunizieren *nicht* direkt miteinander
- evtl. nicht immer verbunden
- evtl. dynamische IP-Adresse

Grundlagen — Peer-to-Peer-Anwendungen

Endsysteme kommunizieren direkt miteinander
- fordern Dienste von anderen Peers an
- nicht permanent verbunden, wechseln dymanisch IP-Adressen
  ⇝ komplexes Management

selbst-skalierend
- neue Peers erhöhen Kapazität, fordern aber auch selber Dienste an

Web und HTTP — Web-Dokumente

Webseiten bestehen aus Basis-HTML-Datei und anderen Objekten (.js, .png,…)
Jedes Objekt über URL (*uniform resource locator*) referenzierbar

HTTP — Überblick

Protokoll der Anwendungsschicht (*hypertext transfer protocol*)
- einfaches, ASCII-basiertes Transferprotokoll

Basiert auf Client/Server-Modell
- *Client*: Browser, der Web-Objekte anfordert, empfängt und darstellt
- *Server*: sendet über HTTP angeforderte Objekte

zwei Nachrichten-Typen: *Request*, *Response*
Zustandslos:
- jeder Request wird individuell bearbeitet
- keine Zustandsinformation auf dem Server

nutzt TCP zur Kommunikation
1. Client initiiert Verbindungsaufbau
2. Server akzeptiert Verbindung
3. Austausch von HTTP-Nachrichten
4. Abbau der TCP-Verbindung

HTTP — Methoden

HTTP-Anfragen können verschiedene Methoden nutzen
GET: Resource von Server zu Client übertragen (z.B. normale Webseite)
POST: Daten zu Ressource übertragen (z.B. Web-Formular)
Weitere Methoden:
- PUT — neue Ressource anlegen
- DELETE — Ressource löschen
- HEAD — wie GET, aber nur HTTP-Header übertragen
- …

HTTP — Status-Codes

Verarbeitungsindikator (Erfolg/Fehlschlag + Gründe)
200: Erfolg; Antwort ist in dieser Nachricht
301: Angefragtes Objekt wurde verschoben (neue URL in Nachricht spezifiziert)
400: Server hat Anfrage nicht verstanden
404: Angefordertes Objekt existiert nicht
505: HTTP-Version nicht unterstützt

HTTP — Verbindungen

Non-persistent HTTP:
- höchstens ein Objekt wird über TCP-Verbindung gesendet, danach geschlossen
  ⇝ Herunterladen mehrerer Objekte erfodert mehrere TCP-Verbindungen

Persistent HTTP: mehrere Objekte über eine TCP-Verbindung

Non-persistent HTTP — Antwortzeit

Round Trip Time (RTT): Zeit, die Paket von Sender zu Empfänger und zurück benötigt
HTTP-Antwortzeit:
- ein RTT für Verbindungsaufbau
- ein RTT für HTTP-Anfrage und erste Antwortbytes
- Zeit t_s für Senden der Datei
  ⇝ **Antwortzeit** $2 * \text{RTT} + t_s$

Cookies

Speichert Nutzer-Server-Zustand
⇝ Server kann Inhalt abhängig von Nutzeridentifikation bereitstellen
Komponenten:
- Cookie-Information in HTTP-Response-Nachricht
- Cookie-Information wird in nachfolgenden HTTP-Requests genutzt
- Datei mit Cookies wird auf Nutzer-Endsystem vom Browser verwaltet
- Datenbank bei Webseite ⇝ Server muss Cookies richtig interpretieren können

Cookies — Privatsphäre

Webseiten unterscheiden Nutzer durch Cookies
⇝ Werbeanbieter können Nutzer über viele Webseiten tracken
Webseiten können durch Cookies sehr viel über Nutzer lernen

Mail — Komponenten

User Agent (UA):
- lesen, senden, weiterleiten
- Beispiele: Outlook, Thunderbird

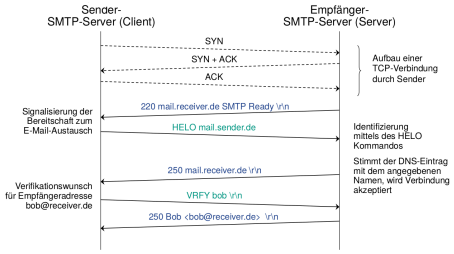
Mailserver:
- *mail transfer agent* (MTA)
- *mail delivery agent* (MDA)
- User-Mailboxen

simple mail transfer protocol (SMTP)
- Client/Server-Modell
- Transfer von Mails vom User Agent zum Mailserver

SMTP — Aufbau

Drei Phasen:
1. Handshake
2. Nachrichtenübermittlung
3. Abschluss

Command/Response-Interaktionen
- ähnlich Request/Response bei HTTP
- Kommandos: ASCII-Text
- Antwort: Statuscode + Nachricht



MIME

Problem: SMTP kann nur ASCII-Texte versenden, keine Dateien
MIME: erweitert Kopfteil einer Nachricht um Formatinformation
- *Content-Type*: Definiert Typ des E-Mail-Inhalts

Mail — Postfach-Abfrage

POP3 (*post office protocol 3*):
- Client holt von Mailserver empfangene/gespeicherte Nachrichten ab
- einfache Funktionalität
- verwaltet Nachrichten im UA, *keine* Synchronisation zwischen mehreren UAs

IMAP (*interactive mail access protocol*):
- Nachtichten werden zentral auf Mailserver verwaltet
- erweiterte Kommandos (Ordner, Filter)

XMPP

Echtzeit-XML-Streaming-Protokoll
Grundlage für Whatsapp usw.
Dezentral, ähnlich wie E-Mail
Clients: zu ihrem jeweiligen Server verbunden
Server: verbinden sich untereinander zur Nachrichtenübermittlung
Adressformat:
- *Nutzer*: Server + Username, z.B. alice@jabber.org
- *Clients*: pro Nutzer, z.B. alice@jabber.org/laptop

DNS — Grundlagen

Ziel: Verwendung von Namen statt IPs
Aufgabe: Zuordnung IP-Adresse ↔ Name
Funktionalitäten:
- *Registrierung* von Namen + IP-Adressen
- *Auflösung* von Namen in IP-Adressen

DNS — Aufbau

Verteilte Datenbank von Name-Servern (DNS-Servern)
- Client-Server-Modell
- Server kann Anfrage an weitere Server weiterleiten

Protokoll der Anwendungsschicht
- Über Port 53 (UDP) realisiert

Basisdienst ⇝ keine Anwendung
- Komplexität am Rande des Netzes lokalisiert
  → Internet-Design-Philoshopie!

DNS — Anfragen

Rekursiv: kennt angefragter Server Antwort nicht, fragt dieser dahinterliegende Server, bis er Antwort bekommt
Iterativ: kennt angefragter Server Antwort nicht, fragt *Client* andere Server
Üblich: Client fragt lokalen Name-Server rekursiv, dieser dann iterativ

DNS — Resource Records (RR)

DNS ordnet Domänen zu Einträgen zu
A / AAAA (Adress): Abbildung Name auf IPv4/IPv6-Adresse
MX (Mail Exchange): Mailserver einer Domäne
NS (Name Server): Nameserver einer Domäne
CNAME (Canonical Name): Alias-Namen für Rechner/Domänen
PTR (Pointer): Abbildung IP-Adresse auf Name