IC4D, Learning Task: Netzwerke
Chat-Kommunikation über einen Relay-Server

Netzwerke

Skript

Cyril Wendl

Inhaltsverzeichnis

Kapitel 1

Netzwerke

1.1 Netzwerke und das Internet

Bevor wir uns mit dem Internet als spezielles Netzwerk befassen, wollen wir allgemein nochmals auf Netzwerke eingehen. Im Alltag treffen Sie Netzwerke an unterschiedlichsten Orten an, zum Beispiel:

- Transport-Netzwerke: Die Post, DHL, Planzer, etc.
- Soziale Netzwerke: TikTok, Youtube, Instagram, etc.
- Mobilitäts-Netzwerke: Das Strassennetz, SBB, Flixbus, etc.
- Shopping-Netzwerke: Aliexpress, Uber, etc.
- Computer-Netzwerke: **Das Internet**, Firmen-Netzwerke etc.

All diese Netzwerke können als Graphen visualisiert und veranschaulicht werden. Gemeinsam ist allen Netzwerken, dass die einzelnen Knoten ihren „Wert“ (ihren Nutzen) erst durch die Verbindung mit anderen Kanten erhalten. Anders gesagt, mit den Worten des „Erfinders des Internets“, Tim Berners-Lee:

> The web is more a social creation than a technical one.

Häufig werden die Begriffe „Internet“ und „Computer-Netze“, bzw. „Rechner-Netze“ austauschbar verwendet. Dabei ist es jedoch wichtig zu verstehen, dass dies zwei unterschiedliche Konzepte sind: während dem mit dem *Internet* ein höchst komplexes, den gesamten Globus umspannendes Netzwerk von Computern gemeint ist, können lediglich zwei miteinander verbundene Computer bereits ein Rechnernetz bilden. Das Internet wird häufig auch als *Netz von (Rechner-)Netzen* bezeichnet. Es ist daher wichtig, dass wir uns zuerst mit einfacheren Rechnernetzen befassen, da diese die Grundbausteine für das Internet bilden.

1.1.1 Geschichte des Internets

Kommunikation war schon seit jeher ein zentrales Bedürfnis der Menschheit. Mit der Erfindung des Telefons, des Radios und später des Internets hat sich die Art und Weise, wie wir kommunizieren, jedoch grundlegend vereinfacht sowie verändert. Im Folgenden schauen wir kurz die wichtigsten Schritte an, die zur Erfindung des Internets geführt haben.

Die Anfänge: ARPANET und die 1960er Jahre

Die Motivation für die Entwicklung des Advanced Research Projects Agency Network (ARPANET) lag darin, ein robustes, dezentrales Kommunikationsnetzwerk zu schaffen, das auch im Falle

von Ausfällen einzelner Knoten funktionsfähig bleibt. Insbesondere sollte das Netzwerk den Austausch von Informationen zwischen Universitäten und Forschungseinrichtungen ermöglichen und die Zusammenarbeit fördern. Ein weiterer wichtiger Beweggrund war die Suche nach einer Kommunikationsinfrastruktur, die im Kontext des Kalten Krieges auch militärischen Anforderungen an Ausfallsicherheit und Flexibilität genügte.Dieses Netzwerk sollte ursprünglich militärischen Zwecken dienen, fand jedoch schnell Anwendung in der akademischen und Forschungswelt.

- 1969: Die erste Nachricht wird zwischen zwei Computern am ARPANET übertragen.
- 1972: Die erste E-Mail wird verschickt.
- 1973: ARPANET wird international, erste Verbindungen nach Norwegen und Grossbritannien.

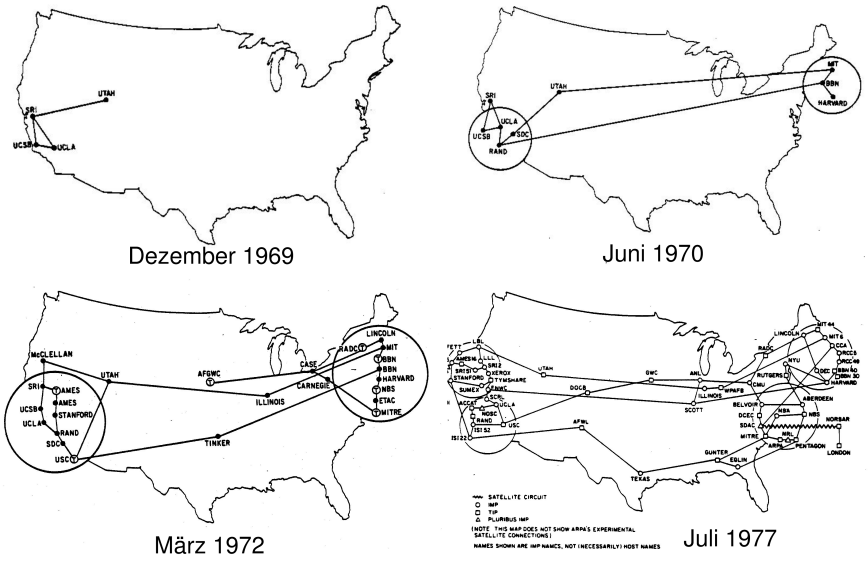


Abbildung 1.1: Ausbreitung des ARPANET von 1969 bis 1977 (Quelle: fibel.org)

Die Entwicklung von Protokollen: TCP/IP

Ein entscheidender Schritt in der Geschichte des Internets war die Entwicklung von Kommunikationsprotokollen, die es verschiedenen Netzwerken ermöglichten, miteinander zu kommunizieren. Insbesondere das Transmission Control Protocol (TCP)/Internet Protocol (IP)-Protokoll, das 1983 als Standard eingeführt wurde, bildete die Grundlage für das moderne Internet.

- 1983: Umstellung des ARPANET auf TCP/IP.
- 1980er: Entstehung weiterer Netzwerke (z.B. CSNET, BITNET) und deren Zusammenschluss.

Eine schematische Darstellung der Verbindung von verschiedenen Netzwerken über das TCP/IP-Protokoll ist in Unterabschnitt 1.2.2 zu finden.

Das WWW und die 1990er Jahre

Die Erfindung des ersten Browsers namens World Wide Web (WWW) durch Tim Berners-Lee im Jahr 1990 revolutionierte die Nutzung des Internets. Kernstück des WWW ist das Hypertext Transfer Protocol (Hypertext Transfer Protocol (HTTP)), das es ermöglicht, Webseiten zu erstellen und diese gegenseitig zu verlinken. Mit der Einführung von HyperText Markup Language (HTML) konnten Inhalte strukturiert und formatiert werden. Dies führte zu einer explosionsartigen Zunahme von Webseiten und Nutzern. Eine typische Webseite, wie wir sie heute kennen, ist über eine Web-Adresse, oder genauer gesagt Uniform Resource Locator (URL) erreichbar, wie in folgendem Beispiel:

```
https://www.beispielseite.ch/unterseite.html
```

Eine URL (Uniform Resource Locator) ist eine eindeutige Adresse, die auf eine Ressource (meistens eine Datei) im Internet verweist. Sie besteht aus mehreren Teilen, darunter das Protokoll (`http://` oder `https://`), der Domain-Name (`www.beispielseite.ch`) und der Pfad zur Datei (`/unterseite.html`).

Der Domain-Name steht für einen Computer im Internet, auf dem die Webseite gespeichert ist. Dieser Computer wird auch als **Webserver** bezeichnet.

Jede Webseite ist dabei als einfache Textdatei gespeichert, mit der Dateiendung `.html` (oder `.htm`). HTML ist dabei eine Art Code-Sprache, die es ermöglicht, Text, Bilder und Links zu strukturieren. Eine einfache HTML-Datei könnte zum Beispiel so aussehen:

```
<!DOCTYPE html>
<html>
<head>
    <title>Beispielseite</title>
</head>
<body>
    <h1>Willkommen auf der Beispielseite</h1>
    <p>Dies ist ein einfacher Textabschnitt.</p>
    <a href="https://www.andereseite.ch/start.html">Andere Webseite</a>
</body>
```

Auf der vorletzten Zeile sehen Sie den Link zu einer anderen Webseite. Dieser Link ist ein sogenannter **Hyperlink**, der es ermöglicht, von einer Webseite zu einer anderen zu navigieren. Hyperlinks sind das Herzstück des WWW und ermöglichen die Vernetzung von Informationen im Internet. Sie bestehen aus einem Text (der anklickbar ist) und einer URL, die die Adresse der verlinkten Seite angibt. Ein Protokoll namens HTTP regelt, wie diese Links angefragt und die entsprechenden Seiten übertragen werden.

Ein Browser kann den HTML-Code benutzerfreundlich anzeigen und Verweise auf andere Webseiten als anklickbare Links anzeigen. Die Kombination von HTML, HTTP und der URL ermöglichte es, Informationen auf einfache Weise zu teilen und zu durchsuchen.

Mit der Einführung von grafischen Webbrowsern wurde das Internet für die breite Öffentlichkeit zugänglich und erlebte einen massiven Popularitätsschub.

- 1991: Das WWW wird der Öffentlichkeit vorgestellt.
- 1993: Der erste grafische Webbrowser (Mosaic) erscheint.
- 1995: Kommerzielle Nutzung des Internets wird möglich, Gründung von Unternehmen wie Amazon und eBay.

Abbildung 1.2: Sir Tim Berners-Lee, der „Erfinder“ des World Wide Web, 2014 (Quelle: Wikipedia)

Das Internet im 21. Jahrhundert

Im 21. Jahrhundert hat das Internet weiterhin rasant an Bedeutung gewonnen. Mit der Verbreitung von schnellem Breitband-Internet, Smartphones und sozialen Medien ist das Internet zu einem unverzichtbaren Bestandteil des täglichen Lebens geworden.

Eine der wichtigsten Erfindungen, um das Internet jederzeit und überall verfügbar zu machen, war das **Mobile Internet**. Mit der Einführung von Smartphones und mobilen Datenverbindungen wurde es möglich, jederzeit auf das Internet zuzugreifen. Dies führte zu einer Explosion von mobilen Anwendungen und Diensten, die das Internet noch zugänglicher machten. Insbesondere die Einführung des iPhones durch die Firma Apple im Jahr 2007 und die damit verbundene Verbreitung von Smartphones haben das Internet revolutioniert. Heute sind Smartphones allgegenwärtig und ermöglichen den Zugriff auf das Internet von nahezu jedem Ort aus.

- 2000er: Verbreitung von schnellem Breitband-Internet, Aufstieg von Google, Facebook, YouTube.
- 2007: Einführung des iPhones, Beginn der Smartphone-Revolution (siehe Video der Produktvorstellung im Jahr 2007).
- 2010er: Mobile Internetnutzung, Smartphones, Internet of Things (IoT).
- Heute: Künstliche Intelligenz, 5G, weltweite Vernetzung.

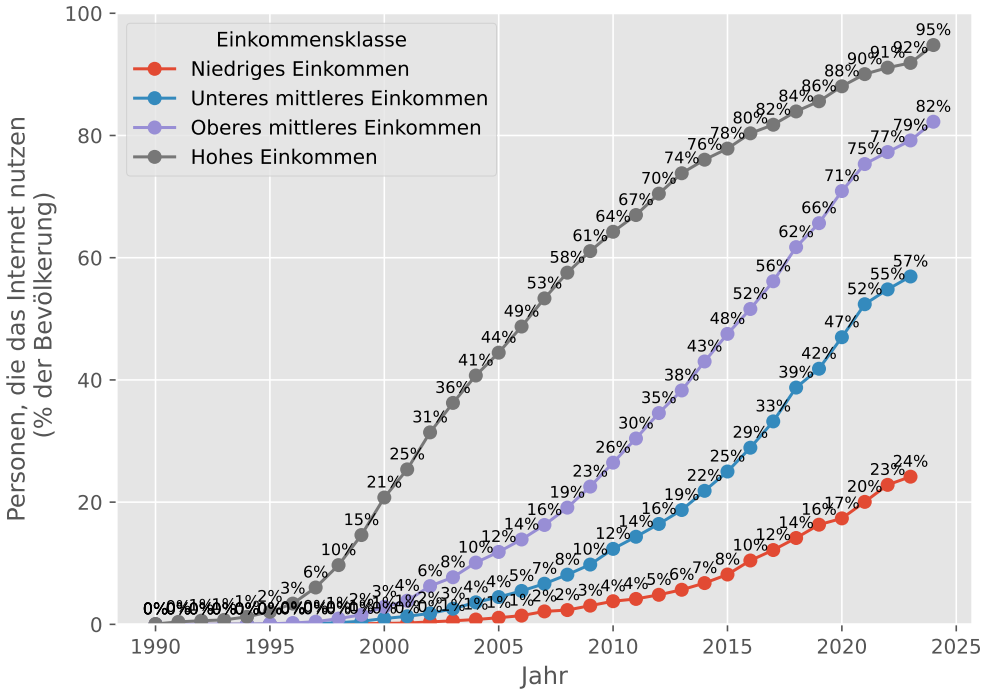


Abbildung 1.3: Entwicklung der weltweiten Internetnutzung nach Einkommensklasse (Quelle: Weltbank-Daten)

Wie auf Abbildung 1.3 zu sehen ist, hat sich die Internetnutzung in den letzten Jahren stark erhöht. Insbesondere in einkommensstarken Ländern ist die Internetnutzung nahezu flächendeckend, während in einkommensschwachen Ländern noch grosse Unterschiede bestehen. Dies wird gut ersichtlich auf der Karte (siehe Abbildung 1.4).

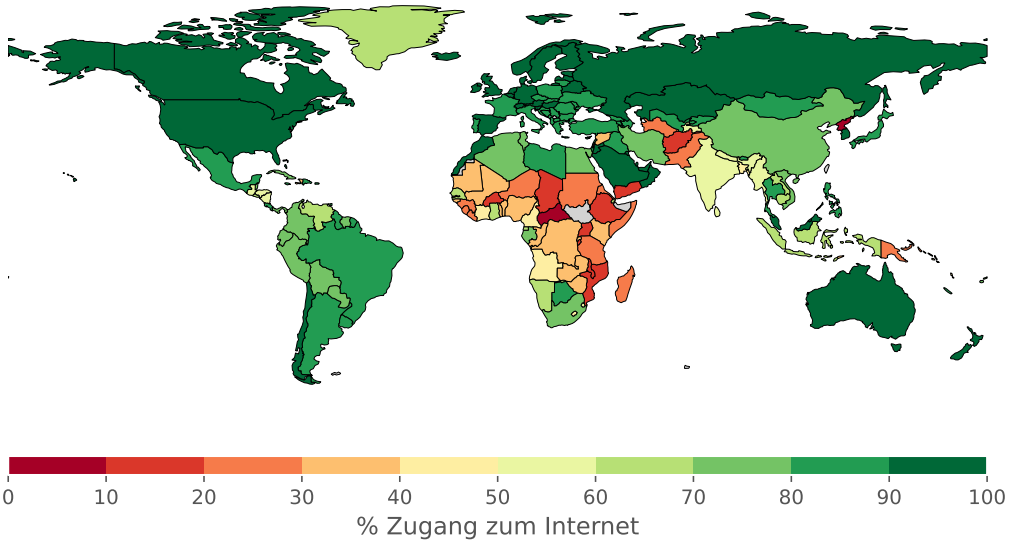


Abbildung 1.4: Internetnutzung nach Land, 2024 (Quelle: Weltbank-Daten)

1.2 TCP / IP und das Schichtenmodell

1.2.1 Schichtenmodell: Analogie

Die Kommunikation über das Internet verläuft in unterschiedlichen Teilschritten. Man spricht von dem *Schichtenmodell*. Als Analogie zum Schichtenmodell sei folgende Situation gegeben, in der die Chefs zweier unterschiedlicher Firmen miteinander kommunizieren wollen (siehe Unterabschnitt 1.2.1):

1. Chef 1 (🚺, in Europa) möchte mit Chef 2 (🚹, in Asien) sprechen.
2. Dazu übermittelt Chef 1 dem Sekretariat Ihre Nachricht an Chef 2. Das Sekretariat schreibt die Nachricht nieder und übermittelt der lokalen Post den Brief an Chef 2.
3. Die lokale Post kümmert sich nun um den Transport zur asiatischen Post, wo der Brief an das Sekretariat von Firma 2 übergeben und schlussendlich an Chef 2 gelangt.

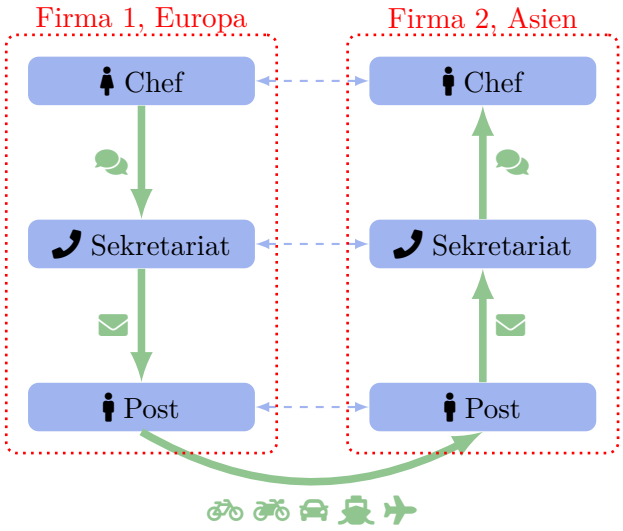


Abbildung 1.5: Analogie zum Schichtenmodell: Kommunikation zwischen zwei grossen Firmen

Dieser Kommunikationsablauf kann in Schichten mit jeweiligen dazugehörigen Prozessen aufgeteilt werden:

1. **Schicht „Chefs“**: damit die beiden Chefs mit einander sinnvolle Gespräche führen können, verfügen Sie über gemeinsame Standards (Sprache, Wissen etc.) sowie Protokolle (standardisierte Abläufe der Begrüssung etc.).
2. **Schicht „Sekretariat“**: Damit die Sekretariate miteinander kommunizieren können, benötigen auch sie standardisierte Abläufe oder Protokolle, etwa, wie man einen Brief korrekt verfässt, datiert und addressiert.
3. **Schicht „Post“**: Die Post verfügt ebenfalls über standardisierte Protokolle (wie werden Briefe und Pakete von A nach B transportiert? Mit welchen Transportmitteln, über welche Routen?)

Die Vorteile des Schichtenmodells scheinen offensichtlich:

- **Unabhängigkeit der Schichten**: Falls die Post das Transportmittel ändert, müssen die Sekretariate die Briefe nicht neu verfassen.
- **Modularität und Spezialisierung**: Statt das jemand alles tun muss (Chef 1 reist nach Asien zu Chef 2), kann der Prozess für die jeweiligten Beteiligten schnell und effizient abgewickelt werden.

1.2.2 Schichtenmodelle TCP / IP und OSI

Im Zusammenhang mit dem Internet werden häufig zwei Schichtenmodelle gegenübergestellt: Das vollständige, detaillierte Open Systems Interconnection (OSI)-Modell mit 7 Schichten sowie das etwas vereinfachte Transmission Control Protocol (TCP)-Internet Protocol (IP)-Modell mit 4 Schichten. Die beiden Schichten, deren Aufgaben, Informationsformen und Protokolle sind in Unterabschnitt 1.2.2 abgebildet.

#	Schicht (OSI)	Schicht (TCP / IP)	Aufgabe	Informationsform	Begriffe / Protokolle
7	Anwendung *Application*	Anwendung *Application*	Hilft bei der Identifizierung des Clients und synchronisiert die Kommunikation.	Nachricht	HTTP, HTTPS, DNS, SMTP, FTP...
6	Darstellung *Presentation*	Anwendung *Application*	Daten von der Anwendungsschicht werden extrahiert und für die Übertragung in das erforderliche Format gebracht.	Nachricht	TLS, SSL, ASCII, UTF-8, ...
5	Sitzung *Session*	Anwendung *Application*	Stellt Verbindung her, hält diese aufrecht, gewährleistet Authentifizierung und sichert die Sicherheit.	Nachricht (oder verschlüsselte Nachricht)	TLS Handshake
4	Transport *Transport*	Transport *Transport*	Nimmt Dienst von der Vermittlungsschicht und stellt ihn der Anwendungsschicht zur Verfügung.	Segment	TCP, UDP, Ports...
3	Vermittlung *Network*	Internet *Internet*	Übertragung von Daten von einem Host zu einem anderen, der sich in unterschiedlichen Netzwerken befindet.	Paket	IPv4, IPv6, ICMP, Router, Gateway
2	Sicherung *Data Link*	Netzzugriff *Data Link*	Übermittlung von Nachrichten von Knoten zu Knoten.	Frame	ARP, MAC, Switch, Bridge, Ethernet, WiFi, ...
1	Bitübertragung *Physical*	Netzzugriff *Data Link*	Herstellung physischer Verbindungen zwischen Geräten.	Bits	Hub, Repeater, Modem, Kabel...

Tabelle 1.1: Internet-Schichten-Modelle Open Systems Interconnection (OSI) und TCP-IP sowie deren wichtigste Aufgaben, Informationsformen, Geräte und Protokolle

Aufgabe 1.1 Netzwerkspiel

Verwenden Sie die Vorlage aus A, um in der Klasse Nachrichten auszutauschen. Notieren Sie sich dabei:

- Welche Schicht für welche Aufgabe zuständig ist.
- Welche Protokolle und Geräte in den jeweiligen Schichten verwendet würden.

Im Folgenden gehen verwenden wir hauptsächlich das etwas vereinfachte Transmission Control Protocol (TCP)-Internet Protocol (IP)-Modelle und gehen kurz auf jede Schicht ein.

Im echten Internet werden Daten häufig als **Frame** (Paket) verschickt, ähnlich wie im Spiel aus Aufgabe 1.1. Jedes Paket besteht dabei aus einem **Header** (Kopfteil) und einem **Payload** (Nutzteil).

Der Header enthält dabei wichtige Informationen über das Paket, wie zum Beispiel die Quell- und Zieladresse, die Sequenznummer und andere Metadaten. Der Payload enthält die eigentlichen Daten, die übertragen werden sollen (z.B. eine Nachricht, ein Bild, ein Video etc.). Jede Schicht fügt dabei ihren eigenen Header hinzu, bevor das Paket an die nächste Schicht weitergegeben wird. Beim Empfänger werden die Header in umgekehrter Reihenfolge entfernt, um die ursprünglichen Daten wiederherzustellen.

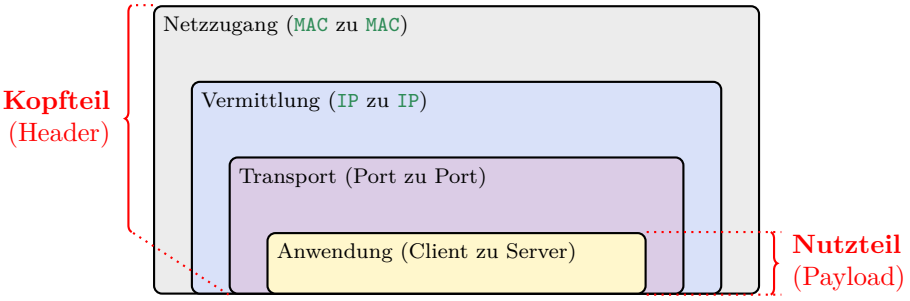


Abbildung 1.6: Illustration eines Datenpakets durch die verschiedenen Schichten des TCP/IP-Modells

1.2.3 Begriffe: Client, Server und Host

Bevor wir uns näher mit den Dimensionen der einzelnen Schichten befassen, sollten wir zuerst einige häufig auftauchende Begriffe klären, von denen im Kontext von Rechner-Netzen häufig die Rede ist.

Definition 1.1 (Client, Server und Host)**:**
Je nach Rolle der Computer in einem Rechner-Netz ist häufig von **Client**, **Server** oder **Host** die Rede.

- Der **Client** $\mathcal{C}$ (von en. *client* = „Kunde“) bezeichnet den Computer $\mathcal{C}$, der etwas von einem anderen Computer $\mathcal{S}$ will. Genauer gesagt handelt es sich nicht um den Computer selber, sondern um ein Programm auf einem Computer $\mathcal{C}$, das etwas anfordert von einem anderen Programm, das auf einem anderen Computer $\mathcal{S}$ läuft. Zur Vereinfachung sprechen wir jedoch häufig einfach von einem Computer $\mathcal{C}$ und einem Computer $\mathcal{S}$.
- Der **Server** $\mathcal{S}$ (von en. *to serve* = „dienen“) bezeichnet den Computer $\mathcal{S}$ (genauer gesagt das Computer-Programm $\mathcal{S}$), das dem Computer $\mathcal{C}$ gibt, was er will. Der Server kann Verbindungsanfragen von anderen Computern akzeptieren oder ablehnen.
- Bei einem **Host** (von englisch *host* = „Wirt“ / „Gastgeber“) ist im allgemeinen ein Computer in einem Netzwerk gemeint, der je nach Situation die Rolle des Servers oder Clients einnehmen kann. Der Name eines Computers in einem Netzwerk wird häufig als **Host Name** bezeichnet.

1.3 Netzzugriff-Schicht

1.3.1 MAC-Adressen

Bisher haben wir mehrheitlich darüber gesprochen, wie Datenpakete zwischen adressierten Empfängern (siehe IP-Adressen) hin- und hergeschickt werden, und wie Addressen in (Sub-)Netzwerke unterteilt und in verständliche Namen umgewandelt werden können. Eine IP-Addresse kann mit der Adresse eines Hauses verglichen werden. Allerdings legt der Briefträger die Briefe und Pakete nicht an einer Adresse ab, sondern in einem Briefkasten, also einem konkreten Objekt. Falls sich der Briefkasten oder sogar das ganze Haus ändert, so bleibt die Adresse doch dieselbe. Ähnlich dieser

Analogie kann in einem Computer die gesamte Hardware ausgetauscht werden, inklusive der Netzwerkkarte, welche sich um das Verschicken und Empfangen von Daten kümmert, und trotzdem sollte der Datentransport zum Computer weiterhin funktionieren. Damit dies klappt, benötigen wir ein weiteres Protokoll, welches die IP-Adresse in eine so genannte Media Access Control (MAC)-Adresse umwandelt.

Definition 1.2 (MAC-Adresse)**:**
Die MAC-Adresse ist die physische Adresse der Netzwerkkarte in einem Computer, also sozusagen der „Fingerabdruck" der Netzwerkkarte. Sie wird normalerweise mit 48 bit (= 6 Bytes) angegeben, welche normalerweise der Kürze wegen in hexadezimaler Schreibweise notiert werden. Ein Beispiel einer MAC-Adresse könnte wie folgt aussehen: `48-2C-6A-1E-59-3D`.

1.3.2 ARP

Damit eine Gerät innerhalb eines lokalen Netzwerks eine Nachricht an ein anderes Gerät senden kann, muss die IP-Adresse des Empfängers in eine MAC-Adresse umgewandelt werden. Da es jedoch keine direkte Möglichkeit gibt, eine MAC-Adresse aus einer IP-Adresse zu berechnen, muss ein spezielles Protokoll verwendet werden, um diese Zuordnung herzustellen. Dieses Protokoll wird als **Address Resolution Protocol** (ARP) (auf Deutsch: *Adressauflösungsprotokoll*) bezeichnet. Dieses wird jedes Mal ausgeführt, wenn ein Computer eine Nachricht an eine bestimmte IP-Adresse senden will, aber die MAC-Adresse des nächsten Empfängers (*gateway*) noch *nicht* kennt. In diesem Fall sendet der Computer eine Anfrage an alle Computer im selben Subnetz (*Broadcast*), um die MAC-Adresse des gewünschten Empfängers zu erfahren. Der Computer mit der gesuchten IP-Adresse antwortet dann mit seiner MAC-Adresse (*Unicast*) und der sendende Computer kann nun die Nachricht an den Empfänger weiterleiten.

Ein Beispiel, wie das ARP-Protokoll funktioniert, könnte wie folgt aussehen:

- Computer `192.168.0.33` will Nachricht senden an `192.168.0.34` im selben Subnetz (Subnetz-Maske: `255.255.255.0`)
- `192.168.0.33` sendet daher eine Nachricht an alle Computer in `192.168.0.x`: „Wer hat die IP `192.168.0.34`"?
- Computer `192.168.0.34` antwortet: „Ich! Meine MAC ist `60:4E:46:F5:08:09`."
- Nun kann der Computer seine Nachricht direkt an den nächsten Empfänger (*gateway*) übergeben.

Die Resultate aller ARP-Anfragen werden in einer so genannten **ARP-Tabelle** gespeichert, damit nicht jedes Mal eine neue Anfrage gesendet werden muss. Diese Tabelle wird periodisch aktualisiert, um sicherzustellen, dass die Zuordnungen aktuell bleiben. Um die ARP-Tabelle auf einem Computer anzuzeigen, kann der Befehl `> arp -a` im Terminal (MacOS, UNIX-Systeme), bzw. `cmd` (Windows) verwendet werden.

1.3.3 Ping

Mit dem Befehl `> ping` kann überprüft werden, ob ein Rechner im Netzwerk erreichbar ist. Dabei wird eine minimale, inhaltslose Nachricht an einen anderen Rechner (=IP-Adresse) geschickt. Das Ziel ist es, zu überprüfen, ob eine gewisse IP-Adresse im Netzwerk existiert. Der Befehl „*ping*" ist vergleichbar mit einem Ping-Pong-Spiel: Der Absender-Rechner, der eine „ping"-Nachricht schickt, erhält vom Ziel-Rechner, sofern dieser erreicht wird, viermal eine „pong"-Nachricht zurückgeschickt. Der `ping`-Befehl kann im Terminal (MacOS, UNIX-Systeme), bzw `cmd` (Windows) wie folgt ausgeführt werden: `> ping [ip address]`, also z.B. `> ping 159.233.1.22`. Der Ping-Befehl ver-

wendet das Internet Control Message Protocol (ICMP)-Protokoll (Internet Control Message Protocol) der Internetschicht, um die Nachrichten zu verschicken. Das ICMP-Protokoll ist ein Protokoll, das speziell für die Übertragung von Kontroll-Nachrichten im Netzwerk entwickelt wurde, allerdings nicht für den Datentransfer von Nutzdaten.

Achtung (Ping of Death)

Eine Spezial-Variante des `ping`-Befehls ist der sogenannte Ping of Death, bei dem ein bösartiger Computer ein zu grosses Datenpaket an einen anderen Computer schickte, das beim Zusammensetzen zum Zusammensturz des Zielrechners führte. Dies ist seit Ende der 90er-Jahre auf den meisten Computern nicht mehr möglich: Die Sicherheitslücke wurde behoben, indem beispielsweise die Paketgrösse vor dem Zusammensetzen überprüft wurde.

Aufgabe 1.2 Filius: Installation

Installieren Sie das Programm Filius, welches wir zur Modellierung von Netzwerken verwenden werden. Die Anleitung dazu finden Sie in Kapitel B.

Aufgabe 1.3 Filius-Anwendung: ARP beobachten

Schauen Sie sich das Filius-Video Nummer 1 an und führen Sie alle gezeigten Schritte ebenfalls in Filius aus.

Ähnlich wie das Datenaustausch-Tool in Filius kann der Datenaustausch in einem echten Netzwerk mit einem Netzwerk-Analyse-Tool wie beispielsweise **WireShark** beobachtet werden. WireShark ermöglicht es, den gesamten Datenverkehr in einem Netzwerk mitzuschneiden und zu analysieren. So können auch die ARP-Anfragen und -Antworten beobachtet werden.

Aufgabe 1.4 WireShark-Auszug

Sie erhalten folgenden WireShark-Auszug eines ARP-Protokolls. Beantworten Sie die Fragen dazu.

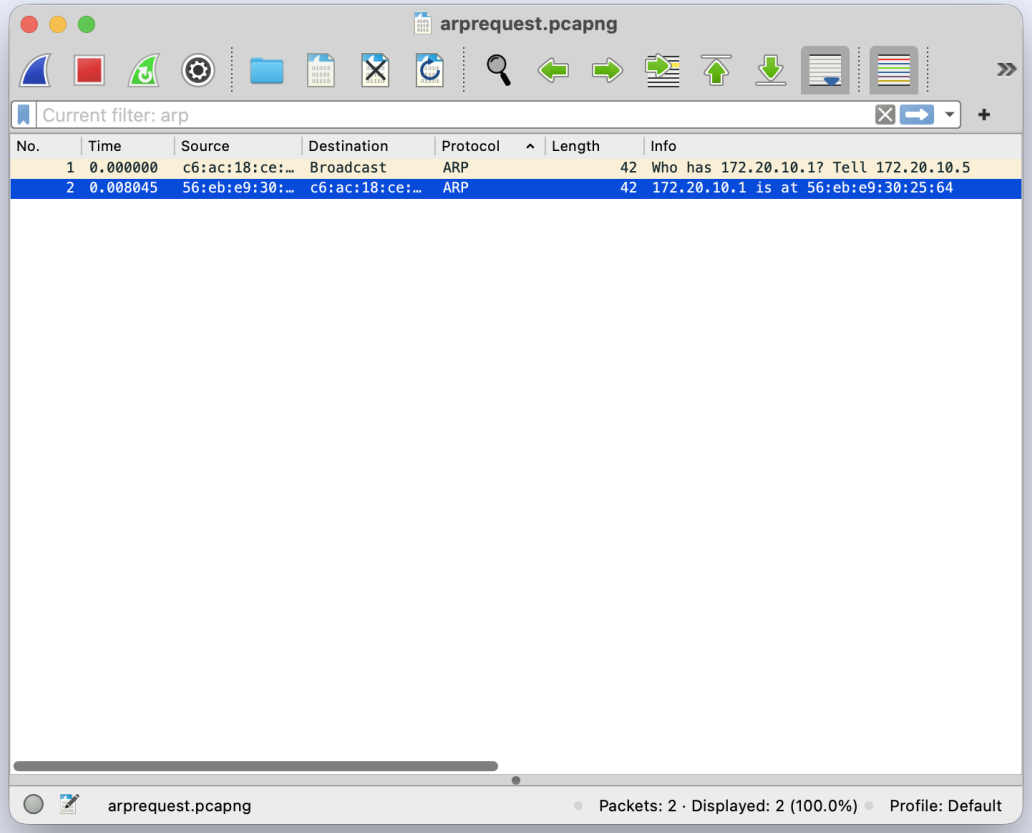


Abbildung 1.7: WireShark-Auszug eines ARP-Protokolls

- Welche IP-Adresse hat der Computer, der die Anfrage sendet?
- Welche MAC-Adresse hat der Computer, der die Anfrage sendet?
- Welche IP-Adresse wird in der Anfrage gesucht?
- Welche MAC-Adresse hat der Computer, der die Antwort sendet?

Aufgabe (Challenge) 1.5 WireShark-Installation und ARP mit WireShark beobachten

Installieren Sie das Netzwerk-Analyse-Tool **WireShark** auf Ihrem Computer (Anleitung siehe wireshark.org). Versuchen Sie, einen ARP-Austausch in Ihrem eigenen Netzwerk zu beobachten. Dazu müssen Sie zuerste den ARP-Cache Ihres Computers leeren. Folgende Befehle müssen in der Befehlszeile eingegeben werden, diese kann wie folgt geöffnet werden:

- Windows: Öffnen Sie die Eingabeaufforderung (cmd) via ⊞ + R und Eingabe von `cmd`
- MacOS: Öffnen Sie das Terminal (z.B. über Spotlight-Suche)

Geben Sie danach folgenden Befehl zum Leeren des ARP-Caches ein:

- Windows: `> arp -d`

- MacOS: `> sudo arp -a -d`

Danach müssen Sie in WireShark die richtige Netzwerkschnittstelle auswählen (z.B. WLAN oder Ethernet) und einen Display-Filter für ARP setzen. Geben Sie dazu in das Filter-Feld Folgendes ein:

```
arp && (arp.src.proto_ipv4 == MEINEIP or arp.dst.proto_ipv4 == MEINEIP)
```

Ihre IP-Adresse können Sie mit dem folgenden Befehl in der Befehlszeile herausfinden:

- Windows: `> ipconfig`
- MacOS: `> ifconfig getifaddr en0`

Starten Sie die Aufnahme in WireShark und versuchen Sie, eine beliebige Webseite in Ihrem Browser zu öffnen. Beobachten Sie die ARP-Anfragen und -Antworten in WireShark.

1.3.4 Verschlüsselung von Netzwerken

In einem lokalen Netzwerk können Datenpakete von jedem Gerät im selben Netzwerk abgefangen und gelesen werden. Um die Sicherheit und Vertraulichkeit der Daten zu gewährleisten, werden verschiedene Verschlüsselungsmethoden verwendet. Eine gängige Methode ist die Verwendung des Wi-Fi Protected Access (WPA)-Protokolls, welches eine starke Verschlüsselung der Datenpakete ermöglicht. WPA verwendet den Advanced Encryption Standard (AES)-Verschlüsselungsalgorithmus, um die Daten zu schützen. Dabei handelt es sich um einen symmetrischen Verschlüsselungsalgorithmus, welcher als sehr sicher gilt und in vielen Anwendungen eingesetzt wird.

Es gibt verschiedene Versionen von WPA, darunter WPA2 und WPA3, die jeweils verbesserte Sicherheitsfunktionen bieten. In einem passwortfreien Netzwerk (offenes Netzwerk) sind die Datenpakete grundsätzlich unverschlüsselt und können von jedem Gerät im selben Netzwerk abgefangen und gelesen werden. Um dies zu illustrieren, schauen wir uns folgenden Wireshark-Auszug eines unverschlüsselten HTTP-Datenpakets an.

Beispiel 1.1 (Wireshark-Auszug eines unverschlüsselten HTTP-Datenpakets)**:**
In Abbildung 1.8 ist ein Wireshark-Auszug eines unverschlüsselten HTTP-Datenpakets zu sehen.

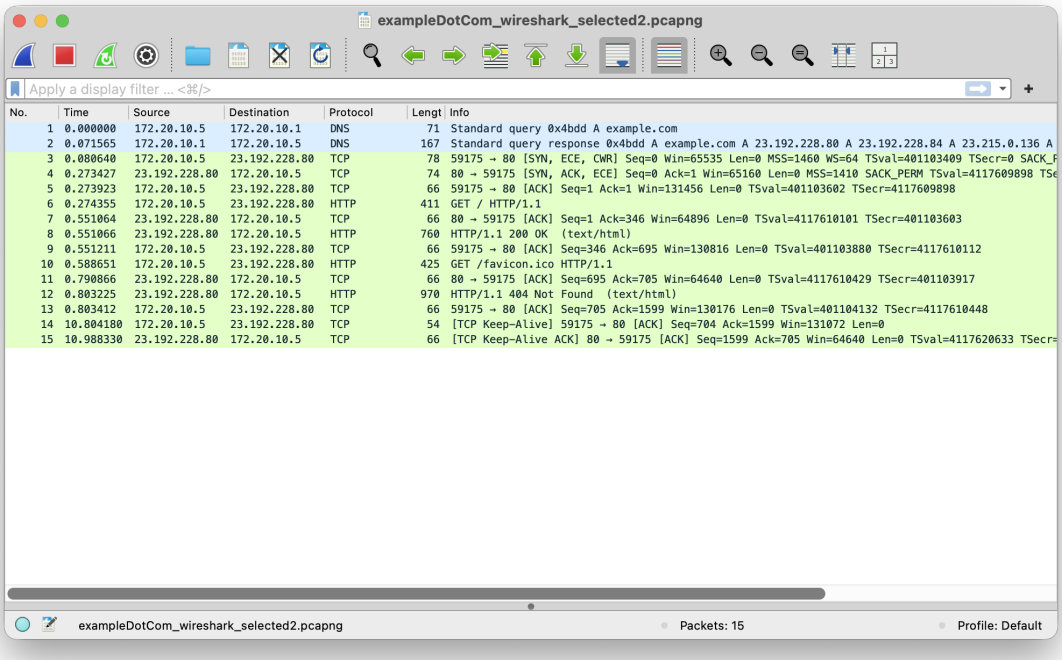


Abbildung 1.8: Wireshark-Screenshot während eines Webseitenaufrufs

Im unteren Teil von Abbildung 1.8 ist der Payload des Pakets dargestellt, welcher den eigentlichen Inhalt der Nachricht enthält. In diesem Fall handelt es sich um eine HTTP-Anfrage, die von einem Client an einen Server gesendet wird. Der Payload enthält dabei Informationen wie die angeforderte Ressource (z.B. eine Webseite), den Hostnamen des Servers und andere Metadaten. Da das Netzwerk unverschlüsselt ist, kann jeder, der Zugriff auf das Netzwerk hat, den Inhalt dieses Pakets lesen und somit die angeforderten Informationen einsehen.

⚠ Achtung (Strafrechtliche Aspekte beim Mithören von Netzwerken)

Das Abfangen und Mitlesen von Datenpaketen in einem Netzwerk ohne ausdrückliche Erlaubnis der beteiligten Parteien ist in vielen Ländern illegal und kann strafrechtlich verfolgt werden. In der Schweiz beispielsweise verstösst das unbefugte Abfangen von Daten gegen das Bundesgesetz über den Datenschutz (DSG) und das Strafgesetzbuch (StGB). Personen, die beim unbefugten Abfangen von Daten erwischt werden, können mit Geldstrafen oder sogar Freiheitsstrafen belegt werden. Es ist daher wichtig, die rechtlichen Rahmenbedingungen zu kennen und einzuhalten.

Aufgabe 1.6 WireShark-Auszug

Sam und Martina haben ein Python-Game geschrieben mit Multiplayer-Funktionalität innerhalb des lokalen Netzwerks. Sam hat das Spiel auf seinem Computer gestartet und Martina möchte sich mit Sams Spiel verbinden. Sams Computer hat die IP-Adresse `192.168.0.97` und Martinas Computer die IP-Adresse `192.168.0.121`. Unten ist ein WireShark-Auszug von Martinas Computer zu sehen, als sie versucht, sich mit Sams Spiel zu verbinden. Beantworten Sie die Fragen dazu.

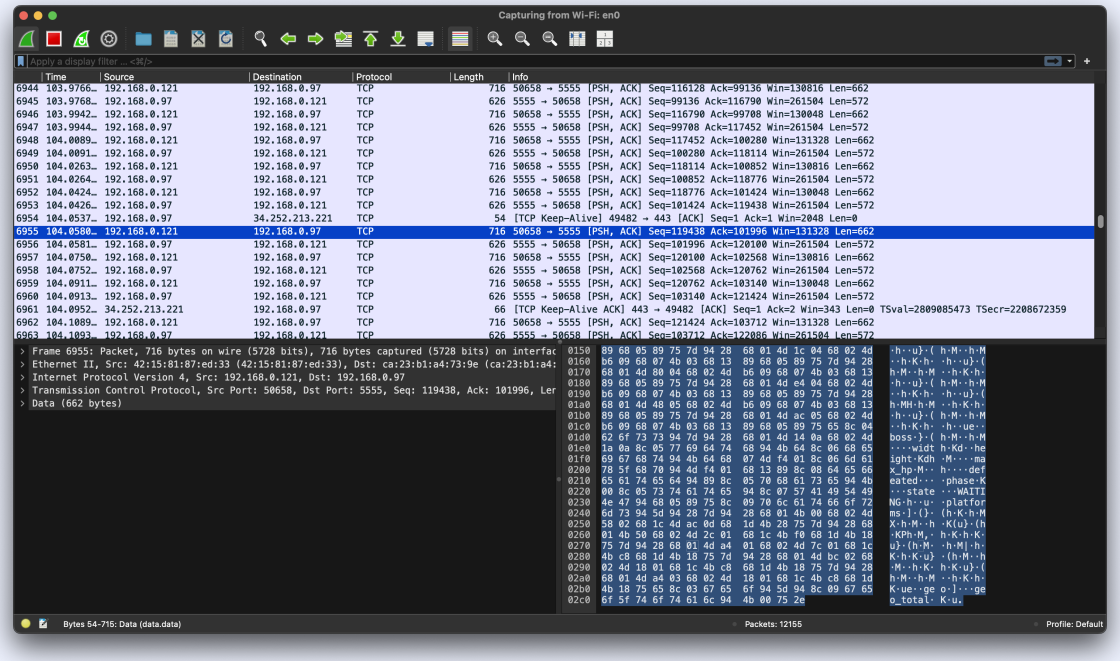


Abbildung 1.9: WireShark-Auszug von Martinas Computer

- Über welches Protokoll versucht Martina, sich mit Sams Spiel zu verbinden? Welche Vorteile bietet dies gegenüber anderen möglichen Protokollen?
- Verwenden Sam und Martina ein unverschlüsseltes oder verschlüsseltes Netzwerk?

1.3.5 Switches

Jede Netzwerkkarte kann nur mit genau einem anderen Gerät in einem lokalen Netzwerk kommunizieren. Damit mehrere Geräte in einem lokalen Netzwerk miteinander kommunizieren können, werden sogenannte **Switches** verwendet. Ein Switch ist ein Gerät, das mehrere Netzwerkkarten miteinander verbindet und es ihnen ermöglicht, Datenpakete direkt aneinander zu senden. Der Switch lernt dabei die MAC-Adressen der angeschlossenen Geräte und leitet die Datenpakete nur an das Gerät weiter, für das sie bestimmt sind. Dadurch wird der Datenverkehr im Netzwerk effizienter gestaltet, da nicht alle Geräte jedes Datenpaket empfangen müssen.

Aufgabe 1.7 Filius-Anwendung: Switch beobachten

Schauen Sie sich das Filius-Video Nummer 2 an und führen Sie alle gezeigten Schritte ebenfalls in Filius aus.

1.3.6 Topologie

Beim Erstellen eines Netzwerks stellt sich als Erstes die Frage, welche Computer man mit welchen anderen Computern verbinden soll. Die Struktur der Verbindungen in einem Netzwerk kann also umformuliert werden als Graphen-Problem, bei dem es zu bestimmen gibt, welche Knoten eines Graphen miteinander über eine Kante verbunden werden. In den folgenden Graphen bezeichnen Knoten die Computer und Kanten die Verbindungen (Kabel) zwischen den Computern. Die verschiedenen möglichen Strukturen eines Netzwerks werden auch als **Topologie** bezeichnet. Einige Beispiel-Topologien sind in Abbildung 1.10 gezeigt.

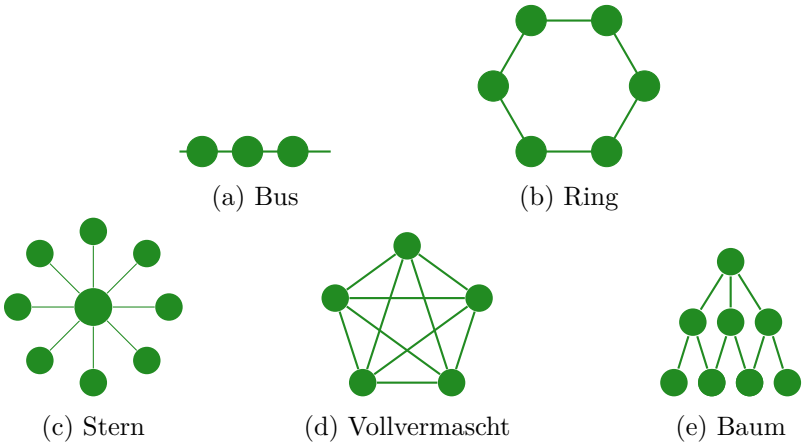


Abbildung 1.10: Beispiele von Netzwerk-Topologien

Da Computer in Realität häufig nur über eine einzige Netzwerkkarte verfügen, können sie auch nur mit einem weiteren Computer verbunden werden. In einem lokalen Netzwerk (LAN) trifft man daher häufig die Form einer Stern-Topologie an, bei der alle Computer mit einer zentralen **Switch** verbunden werden, also einem Gerät, welches mehrere Rechner an einem zentralen Punkt miteinander verbindet.

Aufgabe 1.8 Topologien vergleichen

Vergleichen Sie die Stern-Topologie mit der Vollvermascht-Topologie:

- In welcher Topologie ist es einfacher, ein neues Gerät anzuschliessen?
- In welcher Topologie funktioniert die Kommunikation immer noch gut, selbst wenn einzelne Kabel oder Geräte ausfallen?
- In welcher Topologie müssen mehr Kabel gekauft werden?

Aufgabe 1.9 Kabelanzahl in Stern-Topologien

Wie viele Kabel müssen in der Stern-Topologie gekauft werden…

- Für 6 Geräte?
- Für 10 Geräte?
- Allgemeine Formel?

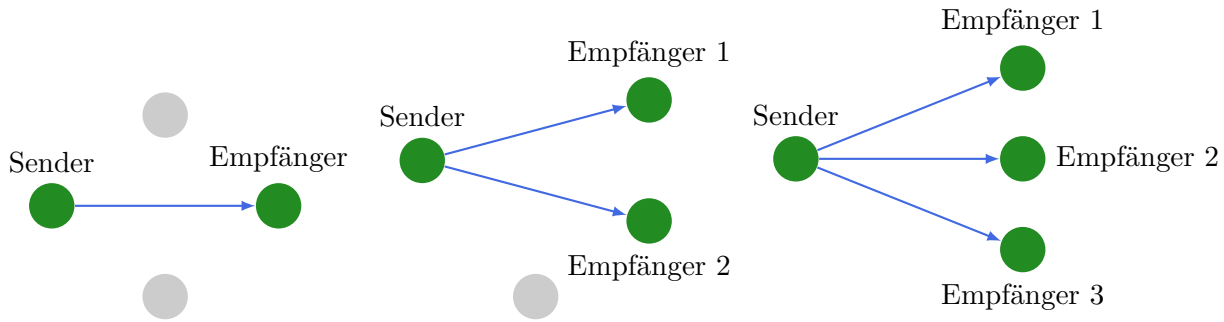
Aufgabe 1.10 Kabelanzahl in Vollvermascht-Topologien

Wie viele Kabel müssen in der Vollvermascht-Topologie gekauft werden…

- Für 6 Geräte?
- Für 10 Geräte?
- Allgemeine Formel?

1.3.7 Unicast, Multicast und Broadcast

Je nachdem für wie viele Teilnehmer eines Netzwerks eine Sendung bestimmt ist, unterscheidet man bei der Datenübertragung zwischen **Unicast**, **Multicast** und **Broadcast** (siehe Abbildung 1.11). Obschon dieses Unterkapitel hier bei der Netzzugriff-Schicht angesiedelt ist, so betrifft diese Unterscheidung alle Schichten des OSI-Modells, da sie die Art der Datenübertragung im gesamten Netzwerk beschreibt.



(a) Unicast: 1:1 (b) Multicast: 1:n (Gruppe) (c) Broadcast: 1:alle

Abbildung 1.11: Unicast, Multicast und Broadcast: Übertragungsarten im Netzwerk

Aufgabe 1.11 Alltagsbeispiele für Übertragungsarten im Netzwerk

Welche Art der Datenübertragung verwenden die folgenden Alltagssituationen?

- Whatsapp-Nachricht an eine befreundete Person
- Lautsprecherdurchsage im gesamten Flughafen
- Lautsprecherdurchsage auf dem Gleis bei der SBB
- Info-Box im Intranet
- Frage einer Person an eine Gruppe der Klasse
- Email der Schulleitung an alle Eltern

1.4 Vermittlungsschicht / Internetschicht

Die Übertragungssicherungsschicht befasst sich mit der stabilen Übertragung von Daten zwischen zwei Rechnern. Wofür sie jedoch nicht zuständig ist, ist die Adressierung und Identifizierung von Computern im Netzwerk. Diesem Aspekt ist die Internetschicht gewidmet.

1.4.1 IP-Adressen

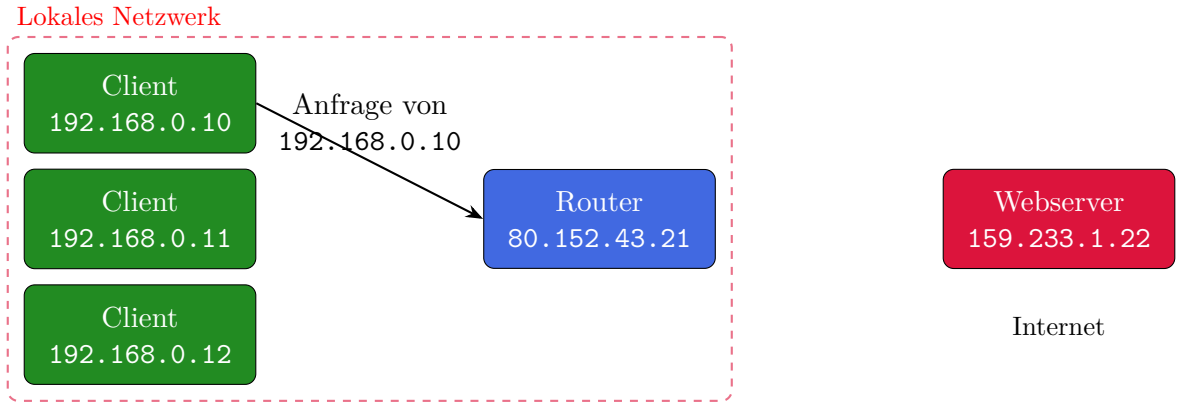
Eine Internet Protocol (IP)-Adresse ist eine Netzwerk-Adresse eines Computers. Somit ist sie in gewisser Weise der Wohnadresse einer Person ähnlich. Häufig werden IP-Adressen noch im **IPv4**-

Format angegeben: Dieses besteht aus 4 bytes, also 4 Zahlen von 0 bis 255. Die IPv4-Adresse wird meist dezimal angegeben (beispielsweise `192.168.0.1`).

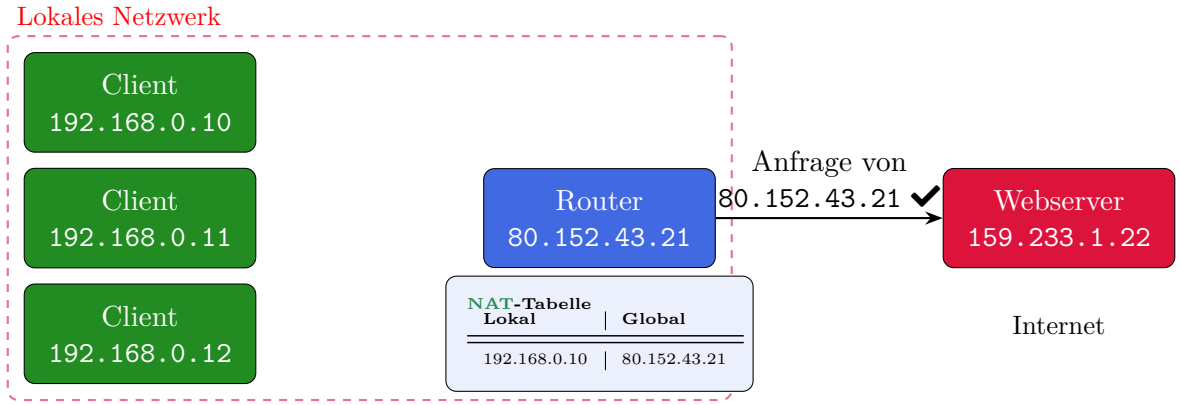
Häufig wird zwischen der **lokalen** und der **globalen** IP-Adresse unterschieden:

- Die **lokale** IP-Adresse (auch private IP-Adresse genannt) ist die Adresse eines Computers innerhalb eines lokalen Netzwerks (z.B. im Heimnetzwerk oder Firmennetzwerk). Diese Adresse ist nur innerhalb dieses lokalen Netzwerks gültig und kann von anderen Computern im Internet nicht direkt erreicht werden.
- Die **globale** IP-Adresse (auch öffentliche IP-Adresse genannt) ist die Adresse eines Computers im gesamten Internet. Diese Adresse ist weltweit eindeutig und ermöglicht es Computern, miteinander über das Internet zu kommunizieren. Die globale IP-Adresse wird in der Regel von einem Internetdienstanbieter (Internet Service Provider (ISP)) zugewiesen.

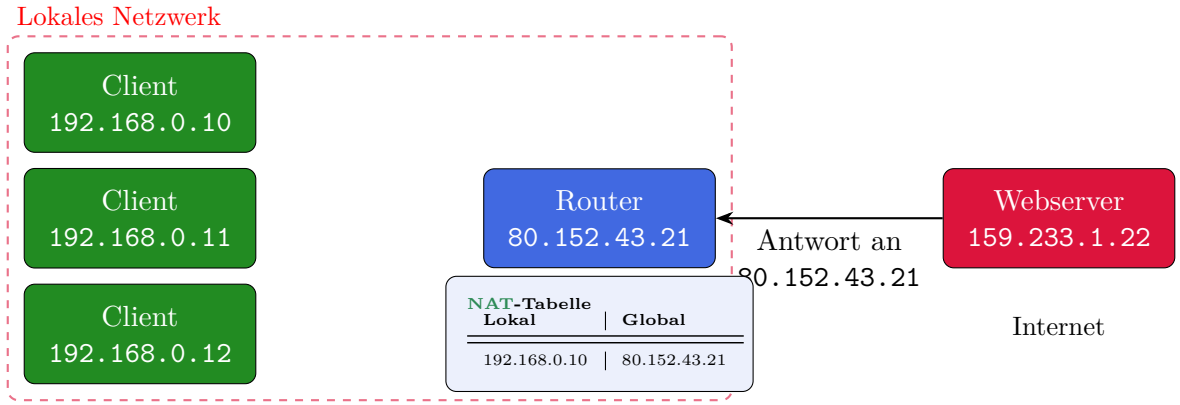
Der Grund, warum es lokale und globale IP-Adressen gibt, liegt darin, dass es nicht genügend globale IP-Adressen für alle Geräte auf der Welt gibt. Durch die Verwendung von lokalen IP-Adressen in privaten Netzwerken können viele Geräte dieselbe lokale Adresse verwenden, solange sie sich in unterschiedlichen Netzwerken befinden. Der Router in einem lokalen Netzwerk übernimmt dann die Aufgabe, die Kommunikation zwischen den lokalen Geräten und dem Internet zu koordinieren, indem er die lokalen Adressen in die globale Adresse übersetzt (dies wird als NAT bezeichnet). Weitere Informationen zu NAT sowie Port Forwarding finden Sie in Kapitel C. Ein Beispiel für die Umwandlung von lokalen in globale IP-Adressen ist in Abbildung 1.12 dargestellt.



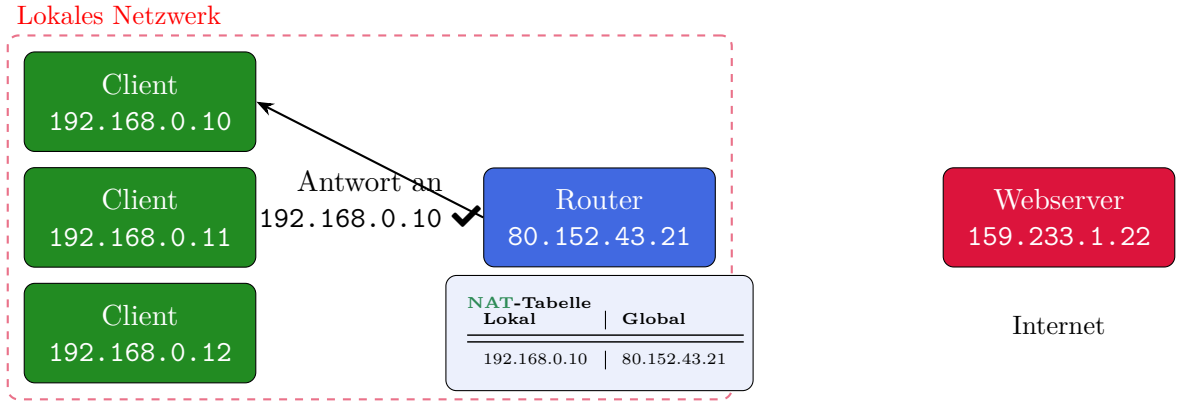
(a) Schritt 1: Client sendet Anfrage



(b) Schritt 2: Router übersetzt lokale IP und sendet Anfrage



(c) Schritt 3: Webserver sendet Antwort



(d) Schritt 4: Router übersetzt globale IP zurück und sendet Antwort

Abbildung 1.12: Beispiel für die Umwandlung von lokalen in globale IP-Adressen mittels NAT

Die lokale Netzwerk-Adresse eines Computers wird häufig automatisch von einem Dynamic Host Configuration Protocol (DHCP)-Server zugewiesen, welcher in der Regel im Router integriert ist. Der DHCP-Server verwaltet dabei einen Pool von verfügbaren IP-Adressen und weist jedem Computer, der sich mit dem Netzwerk verbindet, eine freie Adresse aus diesem Pool zu. Dies geschieht automatisch und vereinfacht die Verwaltung von IP-Adressen in einem Netzwerk erheblich.

Aufgabe 1.12 Eigene lokale und globale IP herausfinden

Finden Sie Ihre eigene lokale IPv4-Adresse heraus.

- **MacOS**: Öffnen Sie das **Terminal** (dieses können Sie via Spotlight-Suche finden: ⌘ + Space). Tippen Sie im Terminal: `> ipconfig getifaddr en0`
- **Windows**: Öffnen Sie die **Eingabeaufforderung** (diese können Sie finden durch Drücken der ⊞-Taste, danach Suche nach dem Programm `cmd` oder Eingabeaufforderung). Tippen Sie in der Eingabeaufforderung: `> ipconfig`

Ihre eigene globale IPv4-Adresse können Sie beispielsweise über die Webseite https://www.whatismyip.com/ herausfinden.

Aufgabe (Challenge) 1.13 `> ping` im lokalen Netzwerk

In vielen Schul-Wireless Local Area Networks (WLANs) ist das Anpingen von anderen Geräten aus Sicherheitsgründen deaktiviert. Wechseln Sie daher in ein anderes Netzwerk, z.B. den Hotspot Ihres Smartphones, um die Übung durchzuführen.

Tauschen Sie zu zweit ihre lokale IPv4-Adresse aus (siehe Aufgabe 1.12) und überprüfen Sie, ob Sie sich gegenseitig im lokalen Netzwerk anpingen können. Verwenden Sie dazu den Befehl `> ping [lokale IP-Adresse des anderen Teilnehmers]` im Terminal (MacOS) oder in der Eingabeaufforderung (Windows).

Aufgabe (Challenge) 1.14

Führen Sie folgendes Skript in Python aus, um zu überprüfen, ob Sie dieselbe lokale und globale IP-Adresse wie in Aufgabe 1.12 herausgefunden haben:

```python
# Gibt die MAC-Adresse, lokale und globale IP-Adresse des Geräts aus.
import uuid
import socket
import requests

# Gibt die MAC-Adresse dieses Geräts aus
mac = uuid.getnode()
mac_hex = ":".join(f"{(mac >> ele) & 0xff:02x}" for ele in range(40, -1, -8)
    )
print(f"{'MAC-Adresse dieses Geräts:':35} {mac_hex}")

# Gibt die lokale IP-Adresse dieses Geräts aus
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.connect(("8.8.8.8", 80))  # Google DNS-Server, kein Datenaustausch
ip_local = s.getsockname()[0]
```

```
s.close()
print(f"{'Lokale IP-Adresse dieses Geräts:':35} {ip_local}")

# Gibt die globale IP-Adresse dieses Geräts aus
ip_global = requests.get("https://api.ipify.org").text
print(f"{'Globale IP-Adresse dieses Geräts:':35} {ip_global}")
```

Aufgabe (Challenge) 1.15 ARP-Scan im lokalen Netzwerk

Falls Sie wissen wollen, welche anderen Geräte im lokalen Netzwerk existieren, können Sie ein ARP-Scan mit dem Programm `arp-scan` durchführen. Am einfachsten kann das Programm `arp-scan` auf MacOS mit Homebrew installiert werden. Führen Sie dazu im Terminal folgenden Befehl aus: `> brew install arp-scan`

Führen Sie danach folgenden Befehl aus, um alle Geräte im lokalen Netzwerk zu scannen: `> sudo arp-scan --localnet --interface=en0`

Die Option `> --localnet` sorgt dafür, dass nur das lokale Netzwerk gescannt wird. Die Option `> --interface=en0` bedeutet, dass die Netzwerkschnittstelle `en0` (normalerweise WLAN) verwendet wird.

Ihre ARP-Tabelle wird danach mit den gefundenen Geräten aktualisiert. Sie können die Tabelle mit dem Befehl `> arp -a` (MacOS) bzw. `> arp -a` (Windows) anzeigen.

Aufgabe 1.16

Bringen Sie folgende IPv4-Adresse in die dezimale Schreibweise:

`11000000.10101000.00000001.00000001`

Aufgabe 1.17

Wie viele IPv4-Adressen gibt es?

Wie wir in Aufgabe 1.17 gesehen haben, gibt es zu wenige Adressen bei IPv4. Daher wird IPv4 graduell durch einen neuen Standard abgelöst, **IPv6**, in welchem jede Adresse über 128 Bits verfügt, womit fast unendlich viele Geräte addressiert werden können.

Definition 1.3 (IPv6-Adressen)**:**
Eine IPv6-Adresse besteht aus 128 Bits, wobei die 128 Bits in 8 Gruppen zu je 16 Bits aufgeteilt werden, welche durch Doppelpunkte getrennt sind. Zur einfacheren Lesbarkeit werden IPv6-Adressen in hexadezimaler Schreibweise angegeben, also mit den Ziffern 0-9 und A-F. Eine typische IPv6-Adresse sieht beispielsweise folgendermassen aus: `2001:0db8:85a3:0000:0000:8a2e:0370:7`
Zur Vereinfachung können führende Nullen in einer Gruppe weggelassen werden, sowie aufeinanderfolgende Gruppen mit dem Wert 0 durch „::“ ersetzt werden. Somit kann die obige Adresse auch als `2001:db8:85a3::8a2e:370:7334` geschrieben werden. Doppelte Doppelpunkt-Kürzungen sind jedoch nicht erlaubt (also z.B. `2001::85a3::8a2e:370:7334` ist ungültig).

Aufgabe 1.18

Wie viele Adressen gibt es bei IPv6?

Aufgabe 1.19

Entscheiden Sie für jede der folgenden Adressen, ob es sich um eine gültige IPv4-Adresse, um eine gültige IPv6-Adresse oder um keine gültige IP-Adresse handelt:

- `256.100.50.25`
- `192.168.1.1`
- `10.0.0`
- `2001:0db8:85a3:0000:0000:8a2e:0370:7334`
- `2001:db8::1`
- `1234:5678:9abc:def0:1234:5678:9abc:def0`
- `2001:db8:::1`
- `192.168.1.1.1`

Aufgabe 1.20

Was ist die grösste mögliche IPv4-Adresse in dezimaler Schreibweise? Was ist die grösste mögliche IPv6-Adresse in hexadezimaler Schreibweise?

Bemerkung 1.1 (IPv4 vs. IPv6)**:**
In der Praxis werden IPv4-Adressen immer noch am häufigsten verwendet, da viele ältere Geräte und Netzwerke nur IPv4 unterstützen. Allerdings wird IPv6 zunehmend wichtiger, da die Anzahl der Geräte im Internet stetig zunimmt und der Adressraum von IPv4 nicht mehr ausreicht. Viele moderne Betriebssysteme und Netzwerke unterstützen bereits IPv6, und es wird erwartet, dass in Zukunft immer mehr Geräte auf IPv6 umsteigen werden. Um den Mangel an IPv4-Adressen zu beheben, werden auch Techniken wie Network Address Translation (NAT), Port Address Translation (PAT) und Port Forwarding eingesetzt, die es ermöglichen, mehrere Geräte hinter einer einzigen öffentlichen IP-Adresse zu betreiben (s. Kapitel C für weitere Informationen zu NAT, PAT und Port Forwarding).

1.4.2 Netzmasken & Subnetze

Subnetzwerke verbinden — wie es der Name sagt — mehrere Hosts zu einem (Sub-)Netzwerk. Dabei gehören mehrere Geräte zum gleichen Subnetz, falls Sie:

- Physisch miteinander verbunden sind (via Kabel, WLAN, etc.)
- Die gleiche Subnetzmaske teilen (später mehr dazu)

Eine (Sub-)Netzmaske hat fast dasselbe Format wie eine IP-Adresse und fasst mehrere IPs zu einer Gruppe, d.h. einem Subnetz zusammen. Die Subnetzmaske gibt an, wie viele Stellen einer IP-Adresse innerhalb eines Subnetzes gleich sein müssen. Sie besteht immer aus zwei IPs-Adressen, wobei die erste IP-Adresse einer gültigen Adresse innerhalb des Subnetzes entspricht, und die zweite IP-Adresse vorgibt, wie stark eine weitere IP-Adresse von der ersten IP-Adresse abweichen dürfte, damit sie immer noch zum selben Netz gehört.

Beispiel 1.2 (Subnetzmasken)**:**
Gegeben sei folgende Subnetzmaske:

	Dezimal	Binär
IP-Adresse von Computer 1	`159.233.1.22`	`10011111.11101001.00000001.00010110`
IP-Adresse von Computer 2	`159.233.1.1`	`10011111.11101001.00000001.00000001`
Subnetzmaske	`255.255.255.0`	`11111111.11111111.11111111.00000000`

Tabelle 1.2: Beispiel einer Subnetzmaske mit zwei IP-Adressen

Um herauszufinden, ob die IP-Adressen von Computer 1 und Computer 2 zum selben Netzwerk gehören, muss überprüft werden, ob die binäre IP-Adresse an all denjenigen Stellen gleich ist, wo die Subnetzmaske = „1“ ist (also an allen grün markierten Stellen in Tabelle 1.2). In diesem Beispiel bedeutet dies, dass alle IP-Adressen, welche mit `159.233.1.[...]` beginnen zum selben Netzwerk dazugehören.

Aufgabe 1.21 Geräte zu Subnetzen zuordnen

Eigene IP	**Netzmaske**	**Ziel-IP**	**Gleiches Subnetz?**
213.45.19.89	255.255.255.0	213.45.17.89	
213.45.19.89	255.255.0.0	213.45.17.89	
88.100.11.17	255.255.255.0	88.100.11.254	
88.100.11.17	0.0.0.0	213.45.19.89	
10.0.0.0	255.255.255.252	10.0.0.1	
1.2.3.0	255.255.255.252	1.2.3.5	

- → IPs dürfen sich nur an denjenigen Stellen unterscheiden, wo in der Maske (binär!) Nullen stehen.
- Umrechner dezimal → binär: https://oinf.ch/interactive/ips-und-netzmaske/

Aufgabe 1.22 Subnetzgrösse berechnen

Typischerweise beginnen alle IP-Adressen, die auf das lokale Netzwerk (=das Netzwerk, in dem sich der eigene Computer befindet) mit `192.168.xxx.xxx`. Die Subnetzmaske lautet also `255.255.0.0`. Wie viele Hosts haben in so einem Netzwerk Platz? Schreiben Sie die Subnetzmaske binär auf und rechnen Sie aus, wie viele unterschiedliche Geräte sich in diesem lokalen Netzwerk befinden können.

Aufgabe 1.23 Subnetzmasken zuordnen

Sechs Subnetze sind gegeben durch je eine Netzmaske und eine IP eines sich darin befindenden Rechners. Entscheiden Sie für jede links aufgeführte IP, zu welchen Subnetzen der entsprechende Rechner gehört.

Netzmaske / **Subnetz**	255.255.255.0		255.255.0.0		255.255.255.240	
IP	3.4.5.6	3.3.3.3	3.4.5.6	3.3.3.3	3.4.5.6	3.3.3.3
3.3.3.4						
4.4.4.3						
3.4.5.99						
3.3.4.4						
3.4.7.7						
3.3.3.17						

Überprüfen: https://oinf.ch/interactive/ips-und-netzmaske/

Definition 1.4 (CIDR-Notation)**:**
Das Netzwerk aus Beispiel 1.2 könnte dezimal angegeben werden wie folgt:

`159.233.1.0/255.255.255.0`

Dabei bezeichnet der erste Teil die IP-Adresse eines (gültigen) Hosts im Subnetz, und der zweite Teil bezeichnet die Subnetzmaske.

Die Subnetzmaske wird häufig verkürzt in der sogenannten Classless Inter-Domain Routing (CIDR)-Schreibweise angegeben. Dabei wird die Subnetzmaske als IP-Adresse gefolgt von einem Schrägstrich und der Anzahl der Einsen in der Subnetzmaske geschrieben. Die Subnetzmaske aus Beispiel 1.2 könnte somit auch als `159.233.1.0/24` geschrieben werden, da die Subnetzmaske 24 Einsen hat.

Durch diese Schreibweise wird zudem schnell ersichtlich, wie gross das Netzwerk maximal sein darf. Die ersten 24 Bits der IP-Adresse müssen immer gleich sein, daher bietet dieses Netzwerk Platz für maximal $2^{32-24} = 256$ unterschiedliche Hosts.

Aufgabe 1.24 CIDR-Notation

Geben Sie die folgenden Subnetzmasken in CIDR-Notation an und notieren Sie die möglichen Adressebereiche für beide Netze 2 und 3, also die kleinste und grösste mögliche IP-Adresse im jeweiligen Subnetz. Notieren Sie zudem, wie viele Hosts in jedem Subnetz Platz haben.

- **Netz 1** (Beispiel): `24.10.10.0` mit Netzmaske `255.255.255.240`
- **Netz 2**: `192.168.1.0` mit Netzmaske `255.255.255.192`
- **Netz 3**: `10.0.0.0` mit Netzmaske `255.128.0.0`

Um die Adressbereiche zu berechnen, können Sie den Online-Rechner unter https://oinf.ch/interactive/ips-und-netzmaske/ verwenden.

	CIDR-Schreibweise	Anzahl Hosts	Adressbereich	
			Start	Ende
Netz 1	24.10.10.0/28	$2^{32-28} = 2^4 = 16$	24.10.10.0	24.10.10.15
Netz 2				
Netz 3				

Tabelle 1.3: Subnetzmasken in CIDR-Notation mit Anzahl Hosts und Adressbereichen

1.4.3 Router & Gateways

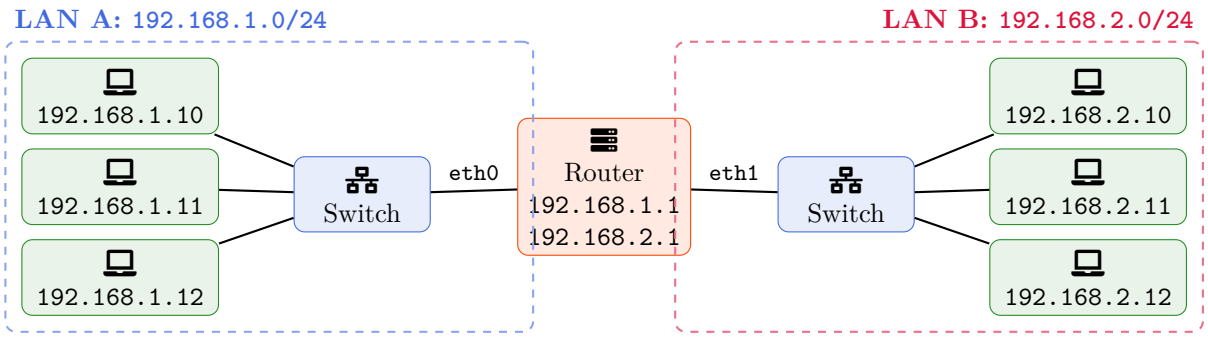


Abbildung 1.13: Zwei Subnetze (LAN A und LAN B) verbunden via Router

Bis anhin haben wir gesehen, wie wir einzelne Rechner mittels einer IP adressieren und wie wir diese mittels einer Subnetze oder und Switches zu einem lokalen Netzwerk oder LAN verbinden. Was, wenn wir jedoch zwei Subnetze zu einem grossen Netzwerk oder WAN verbinden wollen? Genau dies geschieht im Internet, welches im Wesentlichen aus einem Netzwerk von Netzwerken besteht. Um mehrere Netzwerke miteinander zu verbinden, verwenden wir einen **Router**. Diesen schauen wir uns in diesem Abschnitt genauer an.

Ähbnliche einer Haustür verbindet ein Router ein Subnetz (=Haus) mit der Aussenwelt, er befindet sich also an der Grenze zwischen zwei oder mehreren Subnetzen. Über die Routing-Tabelle kann ein Router entscheiden, auf welchen Weg ein Datenpaket an die Aussenwelt verschickt werden kann. Eine typische Routing-Tabelle enthält folgende Informationen:

- **Ziel-IP und Netzmaske**: Diese beiden Spalten definieren, welche Subnetze dem Router bekannt sind (sowohl eigenes Subnetz wie andere Subnetze).
- **Gateway** = Tor, Ein-/Ausfahrt: Der nächstgelegene Zielort, an den ein Paket geschickt werden muss, um eine gewisse Ziel-IP zu erreichen. Das Gateway kann als eine Art „Tor zur Aussenwelt“ angeschaut werden und ist häufig die IP-Adresse des Routers der Ziel-IP. Im Beispiel aus Abbildung 1.13 ist das Gateway für Pakete, die von LAN A zu LAN B geschickt werden sollen, die IP-Adresse des Routers im LAN A, also `192.168.1.1`.
- **Interface** = Schnittstelle: Die physikalische Schnittstelle am Router (z.B. Ethernet-Stecker, daher häufig abgekürzt, beispielsweise, `eth0` für das erste Ethernet-Interface oder `wlan0` für das erste WLAN-Interface), über die eine Ziel-Adresse (Gateway) erreichbar ist. Im Beispiel aus Abbildung 1.13 ist das Interface für Pakete, die von LAN A zu LAN B geschickt werden sollen, das Ethernet-Kabel, das den Router mit dem LAN A verbindet, also `eth0`.

Ein Beispiel einer minimalen Routing-Tabelle ist in Tabelle 1.5 abgebildet.

Ziel-IP	**Netzmaske**	**Gateway**	**Interface**	**Kommentar**
`192.168.1.0`	`255.255.255.0`	`0.0.0.0`	`eth0`	Direkt verbundenes LAN `192.168.1.0/24` wird über Kabel `eth0` erreicht.
`192.168.2.0`	`255.255.255.0`	`0.0.0.0`	`eth1`	Direkt verbundenes LAN `192.168.2.0/24` wird über Kabel `eth1` erreicht.
`0.0.0.0`	`0.0.0.0`	`192.168.1.1`	`eth0`	Standardroute ins Internet (*default gateway*)

Tabelle 1.5: Beispiel einer Routing-Tabelle eines Routers, welcher sich in zwei LANs `192.168.1.0/24` und `192.168.2.0/24` befindet.

Beim sogenannten *default gateway* handelt es sich um die Standardroute, die verwendet wird, wenn keine spezifischere Route für eine Ziel-IP in der Routing-Tabelle gefunden werden kann. In der Regel ist das Default Gateway die IP-Adresse des Routers, der das Subnetz mit dem Internet verbindet.

Aufgabe 1.25 Filius-Anwendung: Router & Gateway

Schauen Sie sich das Filius-Video Nummer 3 an. Laden Sie danach die Datei „`Filius-3.fls`“ von Moodle herunter und bearbeiten Sie die darin enthaltene Netzwerkkonfiguration:

- Notieren Sie zuerst die beiden Netzwerke in CIDR-Notation als Text.
- Konfigurieren Sie danach den Router korrekt, indem Sie die passenden IPs für die beiden Interfaces des Routers vergeben.
- Stellen Sie die Switches als Wolke dar und entfernen Sie die Titel.
- Passen Sie die Gateways aller Clients an.
- Testen Sie die Konfiguration indem Sie einen `ping`-Befehl zwischen zwei Clients ausführen, die sich in unterschiedlichen Netzwerken befinden.

Aufgabe (Challenge) 1.26 Router & Gateway verstehen

Lesen Sie mehr über die Aufgaben und Funktionsweise von Routern und Gateways, indem Sie diese Webseite genau durchlesen.

Bemerkung 1.2 (Automatische Zuweisung von IP-Adressen via DHCP)**:**
In der Praxis ist es unpraktisch, wenn jeder Computer in einem Netzwerk manuell mit einer IP-Adresse konfiguriert werden muss. Daher wird häufig das Dynamic Host Configuration Protocol (DHCP) verwendet, um Computern automatisch eine IP-Adresse zuzuweisen, wenn sie sich mit dem Netzwerk verbinden.

Ein DHCP-Server verwaltet einen Pool von verfügbaren IP-Adressen und weist diese dynamisch an Computer zu, die eine Verbindung zum Netzwerk herstellen. Dies erleichtert die Verwaltung von Netzwerken erheblich, insbesondere in Umgebungen mit vielen Geräten, wie z.B. in Unternehmen oder öffentlichen Netzwerken.

Bemerkung 1.3 (IP-Reservierung)**:**
Falls ein gewisses Gerät immer dieselbe lokale IP-Adresse benötigt (z.B. ein Drucker oder ein Server), kann bei vielen Routern eine so genannte **IP-Reservierung** eingerichtet werden, die sicherstellt, dass dieses Gerät immer dieselbe IP-Adresse erhält, selbst wenn es sich erneut mit dem Netzwerk verbindet. Dies wird typischerweise durch die Zuordnung der MAC-Adresse des Geräts zu einer bestimmten IP-Adresse im DHCP-Server des Routers erreicht. Eine IP-Reservationstabelle könnte beispielsweise wie folgt aussehen:

MAC-Adresse	Reservierte IP-Adresse
`00:1A:2B:3C:4D:5E`	`192.168.1.20`
`11:22:33:44:55:66`	`192.168.1.21`
...	...

Tabelle 1.6: Beispiel einer IP-Reservationstabelle

1.4.4 Zusammenfügen mehrerer LANs zum WAN (Internet)

Stellen Sie sich vor, Sie möchten mehrere lokale Netzwerke (LANs) zu einem grossen Netzwerk (WAN) verbinden, ähnlich wie es im Internet der Fall ist. Jedes LAN soll dabei über einen Router mit dem WAN verbunden werden. Eine kohärente Auswahl von IP-Adressen für jedes Netzwerk und die Router ist dabei entscheidend, um eine übersichtliche und funktionierende Netzwerkinfrastruktur zu gewährleisten.

Aufgabe 1.27 Filius-Anwendung: Router & Gateway

Schauen Sie sich (bei Bedarf) das Filius-Video Nummer 4 an. Schauen Sie sich nach dem Video das etwas komplexere Netzwerk „`Filius 4 Routing Bugs.fls`“ an, welches Sie von Moodle herunterladen können. Leider ist in diesem Netzwerk einiges kaputtgegangen und der Rechner `192.168.0.12` kann den Rechner `68.33.1.11` nicht mehr anpingen. Beheben Sie die Netzwerkkonfiguration, so dass die beiden Rechner sich wieder erreichen können.

- Führen Sie zuerst einen `> ping`-Befehl vom Rechner `192.168.0.12` zum Rechner `68.33.1.11` aus und beobachten Sie anhand der grün aufleuchtenden Striche, bis zu welchem Punkt die Pakete gesendet werden können.
- Setzen Sie bei den Routern korrekte Weiterleitungstabellen ein.
- Setzen Sie bei den Clients korrekte Gateways ein.

1.5 Transportschicht

In der Transportschicht geht es, wie es der Name bereits sagt, darum, wie Daten von einem Host zum nächsten transportiert werden.

Die Daten, die zwischen zwei Computern übertragen werden, werden häufig auch als **Nutzdaten** oder *payload* bezeichnet. Zu diesen Nutzdaten, die zwischen zwei Computern hin- und hergesandt werden sollen, kommen nun noch verschiedene weitere Daten dazu, so genannte **Metadaten**. Metadaten sind dazu zuständig, wichtige Informationen zu den Ursprungsdaten hinzuzufügen, in diesem Falle Daten, um den Transport der Daten zu ermöglichen. Diese Metadaten werden vom Übertragungs-Steuerungs-Protocoll (TCP) zur Verfügung gestellt und genutzt, in einem so genannten *Header* (siehe Abbildung 1.14).

TCP Header	
Source Port	Destination Port
Sequence # (SEQ)	
Acknowledgement # (ACK)	
Weitere TCP-Header-Felder	
Nutzdaten	
... Anwendungsdaten ...	

Abbildung 1.14: TCP-Header (vereinfachte Darstellung)

Einige der wichtigsten Metadaten sind:

- **Der Port** (von engl. *port* = Hafen): Dies ist eine Zahl, mit der der Computer weiss, welche Anwendung (z.B. Mail, Browser, Gaming-Programm etc.) Daten verschickt. Ein Computer kann, wie Sie vermutlich wissen, gleichzeitig über mehrere Programme Daten von anderen Computern empfangen und an sie schicken: sie können beispielsweise gleichzeitig eine Nachricht über WhatsApp schicken und ein Video schauen. Damit der Computer weiss, welche Anwendung die Daten verschickt, verseht er die Daten typischerweise mit einem Port. Obschon Ports für jedes Programm frei gewählt werden können, werden für bekannte Protokolle typischerweise Standard-Portnummern verwendet (siehe Tabelle 1.7).

Port	**Anwendung**	**Verwendung**
20	File Transfer Protocol (FTP)	Dateitransfer
21	File Transfer Protocol (FTP)	Befehlssteuerung
22	Secure Shell (SSH)	Sichere Shell-Zugriffe
25	Simple Mail Transfer Protocol (SMTP)	E-Mail-Versand
53	Domain Name System (DNS)	Namensauflösung
80	Hypertext Transfer Protocol (HTTP)	Webseiten-Abruf
110	Post Office Protocol Version 3 (POP3)	E-Mail-Abholung
143	Internet Message Access Protocol (IMAP)	E-Mail-Management
443	Hypertext Transfer Protocol Secure (HTTPS)	Verschlüsselter Webseiten-Abruf
993	Internet Message Access Protocol (IMAP)	E-Mail-Management über SSL
995	Post Office Protocol Version 3 (POP3)	E-Mail-Abholung über SSL

Tabelle 1.7: Gängige TCP-Ports, Anwendungen und deren Verwendung

  Sowohl der *Client* wie auch der *Server* versehen die Daten mit jeweils einer Zahl, die für den Ursprungs-Port (*Source Port*), respektive den Ziel-Port (*Destination Port*) stehen.
- **Ziffern** für die *Sequence* (Sequenz) und *Acknowledgement* (Anerkennung) von Paketen, die die richtige Reihenfolge der gesendeten und erhaltenen Daten gerantieren.
- Weitere Daten, die für den Transport notwendig sind.

Aufgabe 1.28 Ports für HTTP-Verbindungen

Testen Sie, was beim Aufrufen von `http://www.sbb.ch:80` und `http://www.sbb.ch:81` passiert. Erklären Sie den Unterschied.

Nebst TCP gibt es noch ein weiteres Protokoll in der Transportschicht, das UDP. Im Gegensatz zu

TCP stellt UDP keine zuverlässige Datenübertragung sicher, sondern sendet die Daten einfach los, ohne zu überprüfen, ob sie auch angekommen sind. Dies ist nützlich für Anwendungen, bei denen Geschwindigkeit wichtiger ist als Zuverlässigkeit, wie z.B. bei Online-Gaming oder Video-Streaming, oder auch für Virtual Private Network (VPN)-Verbindungen.

Aufgabe (Challenge) 1.29 Weitere Port-Aufgaben

Lösen Sie die Socket- und Port-Aufgaben auf folgendem Link: https://www.inf-schule.de/rechnernetze/anwendung/socketprogrammierung/Anfragen_an_einen_Server_Stellen.

Zusätzlich zur Steuerung der Datenübertragung können TCP-Metadaten genutzt werden, um die erste Verbindung zwischen zwei Computern herzustellen. Dies erfolgt typischerweise über das 3-Wege-Handschlag-Protokoll (siehe Abbildung 1.15).

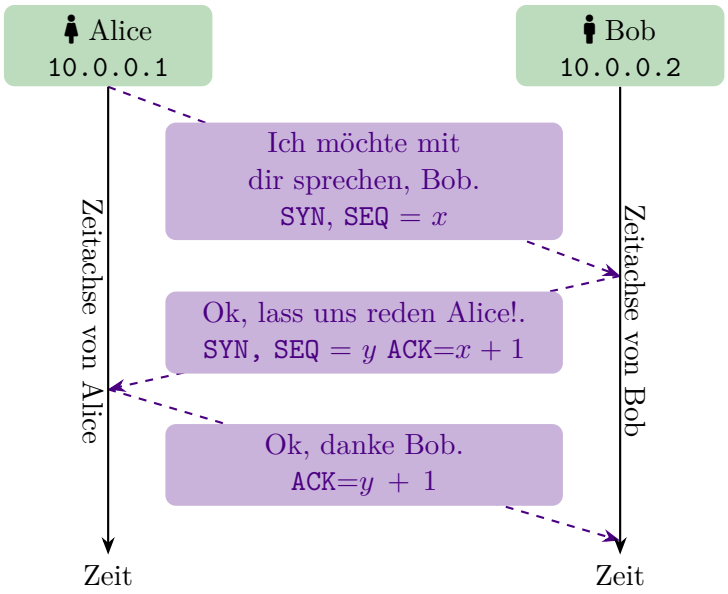


Abbildung 1.15: Drei-Wege-Handschlag im TCP-Protokoll

Folgende Schritte des Drei-Wege-Handschlags werden in Abbildung 1.15 illustriert.

1. **Schritt 1 (`SYN`):** Computer A (*Client*, `10.0.0.1`) möchte eine Verbindung zu Computer B (*Server*, `10.0.0.2`) aufbauen und sendet deshalb ein TCP-Segment mit gesetztem `SYN`-Flag. Damit signalisiert der Client, dass er die Kommunikation starten möchte, und teilt zugleich mit, mit welcher initialen Sequenznummer (`SEQ`) er seine Segmente nummeriert.
2. **Schritt 2 (`SYN + ACK`):** Computer B antwortet auf die Anfrage mit einem Segment, bei dem `SYN` und `ACK` gesetzt sind. Das `ACK`-Flag bestätigt den Empfang des vorherigen Segments von Computer A (inklusive dessen Sequenznummer plus 1), und das `SYN`-Flag gibt an, mit welcher eigenen initialen Sequenznummer der Server seine Segmente starten wird.
3. **Schritt 3 (`ACK`):** Computer A bestätigt diese Antwort abschliessend mit einem Segment mit gesetztem `ACK`-Flag. Damit ist der Verbindungsaufbau abgeschlossen: Beide Computer haben die Start-Sequenznummern synchronisiert und können nun über eine zuverlässige TCP-Verbindung mit dem eigentlichen Datentransfer beginnen.

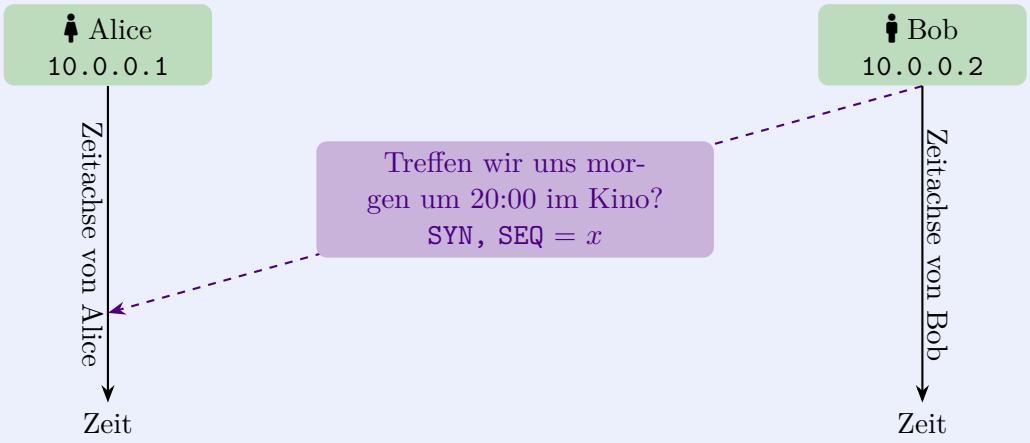
Somit ist die TCP-Verbindung ist nun erstellt und die beiden Maschinen können miteinander kommunizieren. Die Sequenznummern werden im Folgenden immer nach folgender Logik erhöht:

- Für jedes gesendete Byte werden die Sequenznummern um 1 erhöht.

- Für jedes empfangene Byte wird die Acknowledgement-Nummer um 1 erhöht.
- Spezielle TCP-Flags wie `SYN` und `FIN` erhöhen die Sequenznummer jeweils um 1.

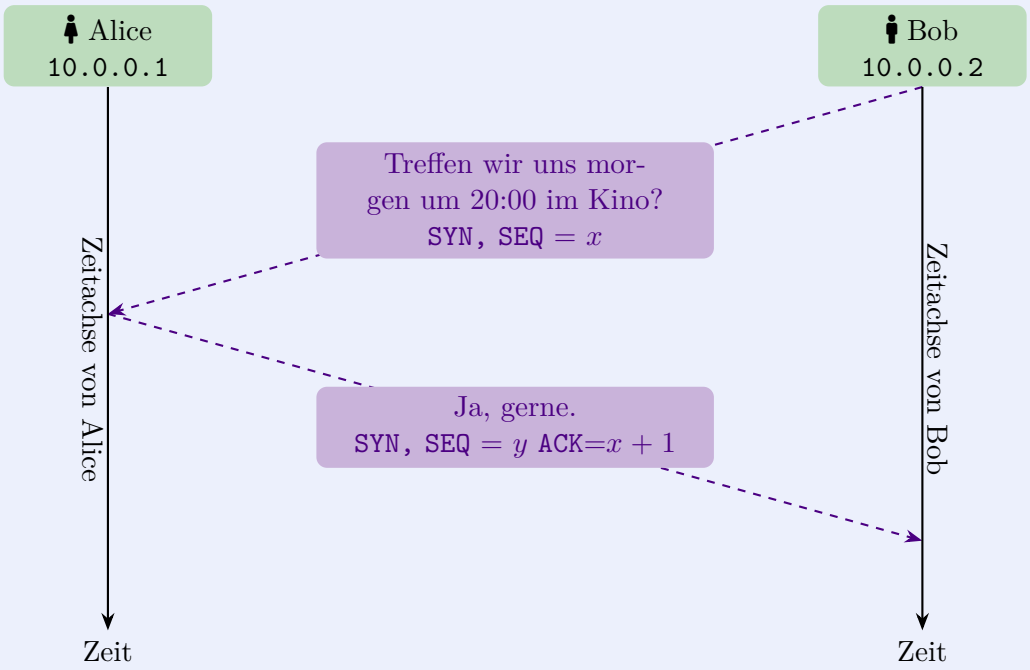
Aufgabe 1.30 Fragen zum Drei-Wege-Handschlag [1]

Erklären Sie, weshalb es in folgendem Beispiel nicht sinnvoll wäre, dass Bob morgen ins Kino ginge.



Aufgabe 1.31 Fragen zum Drei-Wege-Handschlag [1]

Erklären Sie, weshalb es in folgendem Beispiel nicht sinnvoll wäre, dass Alice morgen ins Kino ginge.



1.6 Anwendungsschicht

1.6.1 Echo-Server

Ein einfacher Anwendungsfall eines Servers ist der sogenannte **Echo-Server**. Ein Echo-Server empfängt Nachrichten von einem Client und sendet diese unverändert zurück. Dies kann nützlich sein, um die Erreichbarkeit eines Servers zu testen oder um die Latenzzeit zwischen Client und Server zu messen. Ein Echo-Server arbeitet typischerweise auf einem bestimmten Port.

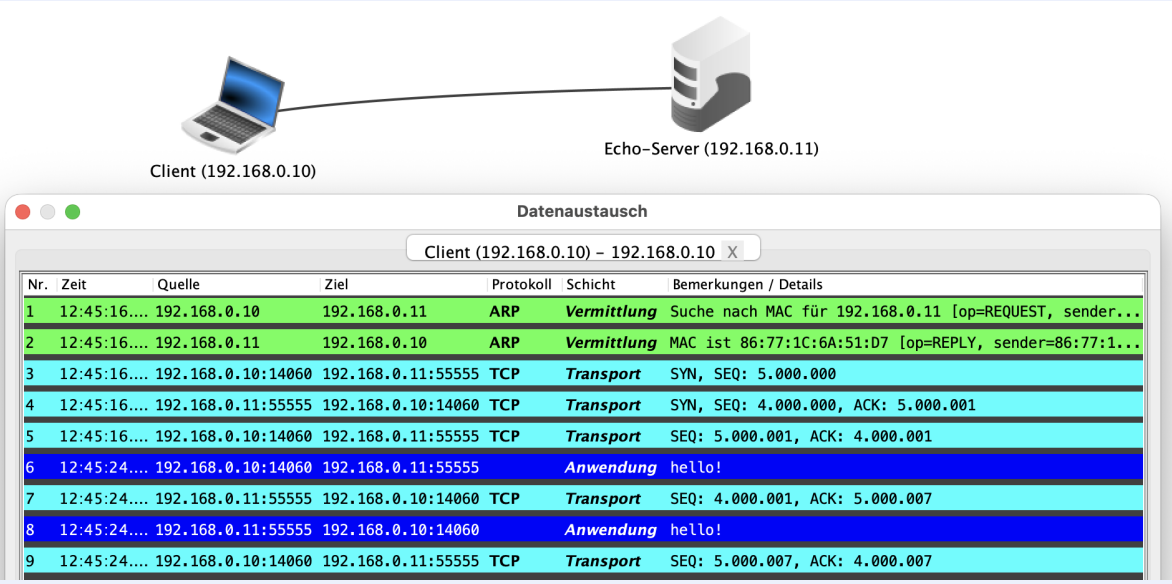
Aufgabe 1.32 Filius-Anwendung: Echo-Server

Schauen Sie sich das Filius-Video Nummer 5 an. Erstellen Sie selber einen Echo-Server sowie zwei Clients in Filius, die mit dem Echo-Server kommunizieren können. Testen Sie die Konfiguration, indem Sie Nachrichten von den Clients an den Echo-Server senden und überprüfen, ob die Nachrichten korrekt zurückgesendet werden.

Aufgabe 1.33

Schauen Sie sich das Datenaustausch-Fenster von Aufgabe 1.32 genauer an.

1. Erklären Sie, was auf jeder Zeile passiert.
2. Über welche Ports erfolgt die Kommunikation beim Server und beim Client?



Nr.	Zeit	Quelle	Ziel	Protokoll	Schicht	Bemerkungen / Details
1	12:45:16....	192.168.0.10	192.168.0.11	ARP	*Vermittlung*	Suche nach MAC für 192.168.0.11 [op=REQUEST, sender...
2	12:45:16....	192.168.0.11	192.168.0.10	ARP	*Vermittlung*	MAC ist 86:77:1C:6A:51:D7 [op=REPLY, sender=86:77:1...
3	12:45:16....	192.168.0.10:14060	192.168.0.11:55555	TCP	*Transport*	SYN, SEQ: 5.000.000
4	12:45:16....	192.168.0.11:55555	192.168.0.10:14060	TCP	*Transport*	SYN, SEQ: 4.000.000, ACK: 5.000.001
5	12:45:16....	192.168.0.10:14060	192.168.0.11:55555	TCP	*Transport*	SEQ: 5.000.001, ACK: 4.000.001
6	12:45:24....	192.168.0.10:14060	192.168.0.11:55555		*Anwendung*	hello!
7	12:45:24....	192.168.0.11:55555	192.168.0.10:14060	TCP	*Transport*	SEQ: 4.000.001, ACK: 5.000.007
8	12:45:24....	192.168.0.11:55555	192.168.0.10:14060		*Anwendung*	hello!
9	12:45:24....	192.168.0.10:14060	192.168.0.11:55555	TCP	*Transport*	SEQ: 5.000.007, ACK: 4.000.007

Abbildung 1.16: Datenaustausch-Fenster eines Echo-Servers in Filius

1.6.2 Email-Server

Ein Echo-Server ist zugegebenermassen ein sehr einfacher Anwendungsfall. In der Praxis gibt es viele komplexere Server, die verschiedene Dienste bereitstellen. Ein häufig verwendeter Server ist der **Email-Server**. Ein Email-Server ist dafür verantwortlich, E-Mails zu empfangen, zu speichern und zu senden. Es gibt verschiedene Protokolle, die für den Betrieb von Email-Servern verwendet werden, darunter:

- **Simple Mail Transfer Protocol (SMTP)**: Dieses Protokoll wird hauptsächlich zum Senden

von E-Mails verwendet. Es ermöglicht es einem Client, eine E-Mail an einen Server zu senden, der die E-Mail dann an den Empfänger weiterleitet. Standardmässig wird Simple Mail Transfer Protocol (SMTP) über den Port 25 betrieben.

- **Post Office Protocol Version 3 (POP3)**: Dieses Protokoll wird verwendet, um E-Mails von einem Server herunterzuladen. Es ermöglicht es einem Client, E-Mails vom Server abzurufen und lokal zu speichern. Standardmässig wird Post Office Protocol Version 3 (POP3) über den Port 110 betrieben.
- **Internet Message Access Protocol (IMAP)**: Dieses Protokoll wird ebenfalls zum Abrufen von E-Mails verwendet, bietet jedoch mehr Funktionen als POP3. Mit Internet Message Access Protocol (IMAP) können Clients E-Mails auf dem Server verwalten, ohne sie herunterzuladen. Standardmässig wird IMAP über den Port 143 betrieben.

Aufgabe (Challenge) 1.34 Filius-Anwendung: Email-Server

Erstellen Sie in Filius eine Client-Server-Architektur, ähnlich derjenigen aus Video 5. Diesmal soll der Server jedoch als Email-Server fungieren, der sowohl SMTP als auch POP3 unterstützt. Installieren Sie dazu auf dem Server einen „E-Mail-Server“ und auf dem Client einen „E-Mail-Client“. Indem Sie den E-Mail-Server konfigurieren, richten Sie automatich sowohl den SMTP- wie auch den POP3-Dienst ein. Zunächst müssen Sie auf dem Server alle Konten anlegen, die es geben soll, und die Maildomain festlegen. Heisst die Maildomain beispielsweise `filius.ch`, so enden alle Email-Adressen auf `@filius.ch`. Auf der Client-Seite muss ebenfalls zuerst ein Konto eingerichtet werden. Testen Sie die Konfiguration, indem Sie eine E-Mail von einem Client an einen anderen Client senden und überprüfen, ob die E-Mail korrekt empfangen wird.

Weshalb braucht es überhaupt einen Email-Server? Ohne einen Email-Server müsste jeder Computer, der eine E-Mail senden oder empfangen möchte, direkt mit dem Computer des Empfängers kommunizieren. Dies wäre jedoch unpraktisch, insbesondere wenn der Empfänger offline ist – in diesem Falle müsste der Sender warten, bis der Empfänger wieder online ist, um die E-Mail zuzustellen, oder die Mail würde ungelesen im Internet verloren gehen, da der Empfänger nicht erreichbar ist. Im Allgemeinen sind Servers rund um die Uhr online. Im Beispiel eines Email-Servers kann der Server die E-Mails speichern, bis der Empfänger sie abruft. Ähnlich ist es auch bei anderen Servern, wie beispielsweise Web-Servern, welche wir als nächstes betrachten.

1.6.3 Web-Server

Eine Ihnen bekannte Situation findet sich vor, wenn ein Client $\mathcal{C}$ eine Webseite von einem Server $\mathcal{S}$ anfordert. Dies geschieht beispielsweise, wenn Sie von Ihrem Mobiltelefon oder Laptop aus eine Webseite über den Browser aufrufen. Im Folgenden werden wir auf vereinfachte Weise veranschaulichen, welche Schritte geschehen müssen, damit Sie ihre Webseite sehen können.

1. Als erstes rufen Sie Ihren **Browser** (von en. *to browse* = „stöbern“) auf, dies können beispielsweise sein: (a) Chrome (b) Firefox (c) Safari (d) Edge (e) etc. Dabei handelt es sich um Programme, die eine Datei (meist `.html`) aus dem Internet anfordern und diese nach dem Empfang darstellen können. Die angeforderten Dateien enthalten dabei Informationen zu Inhalten, Struktur, Darstellung und Interaktivität einer Webseite. Das Senden / Empfangen von Webseite-Dateien selber übernimmt das Betriebssystem (z.B. Windows, MacOS), nachdem es durch den Browser dazu aufgefordert wurde.
2. Sobald Sie Ihren Browser aufgerufen haben, geben Sie eine Web-Adresse ein. Eine solche

Web-Adresse wird im Allgemeinen URL genannt und besteht aus folgenden Komponenten:

`http://` (Protokoll) `www.` (Server) `beispiel.` (Domain) `ch` (TLD) `/dokumente/` (Ordner) `reglemente.html` (Dateiname)

- Der erste Teil (in diesem Fall `http://`) bestimmt das Protokoll, mit welchem die Datei vom Server angefordert wird.
- Der zweite Teil (`www.beispiel.ch`) bezeichnet dabei den *Server* $\mathcal{S}$, von dem wir die Datei anfordern. Dieser Name entspricht in der Regel einer einfachen IP-Adresse, wie beispielsweise `192.168.10.2`. Die Endung (`.ch`) steht für eine sogenannte Top-Level Domain (TLD), in diesem Fall wird durch „`.ch`“ eine Schweizer Webseite bezeichnet.
- Alles nach der TLD ist ein gewöhnlicher Ordnerpfad auf dem Computer, der schlussendlich zur Datei „`reglemente.html`“ führt, die den Inhalt der Webseite enthält

Der schematische Ablauf eines Webseiten-Abrufs mit dem HTTP-Protokoll ist in Abbildung 1.17 abgebildet.

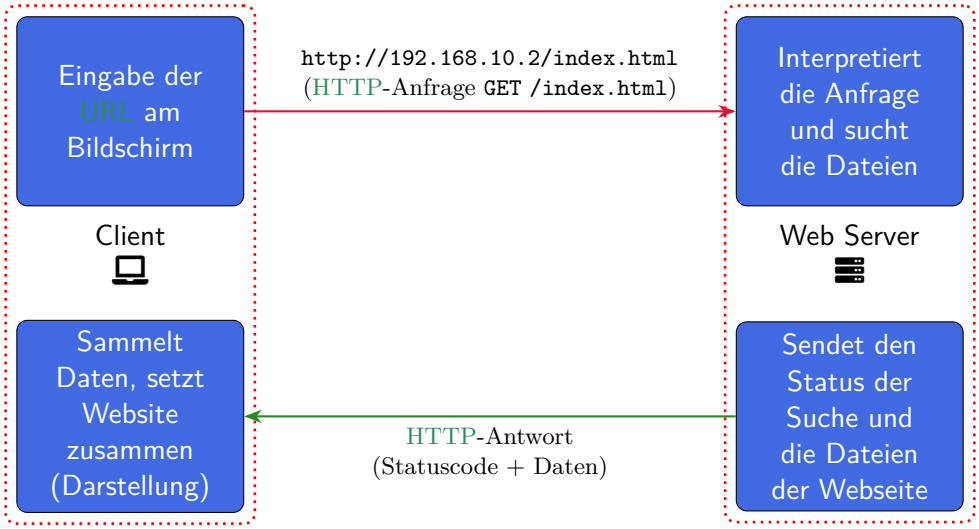


Abbildung 1.17: Schematischer Ablauf eines Webseiten-Aufrufs mit dem HTTP-Protokoll

Wir können den Ablauf einer Abfrage etwas detaillierter aufzeigen, indem wir eine weitere Stufe für den End-Benutzer des Clients einfügen. Hierdurch wird ersichtlich, dass der Browser sich um die Kommunikation mit dem Server kümmert, indem er das HTTP-Protokoll verwendet, um Datei-Anfragen an den Server zu schicken und um diese zu erhalten (siehe Abbildung 1.18).

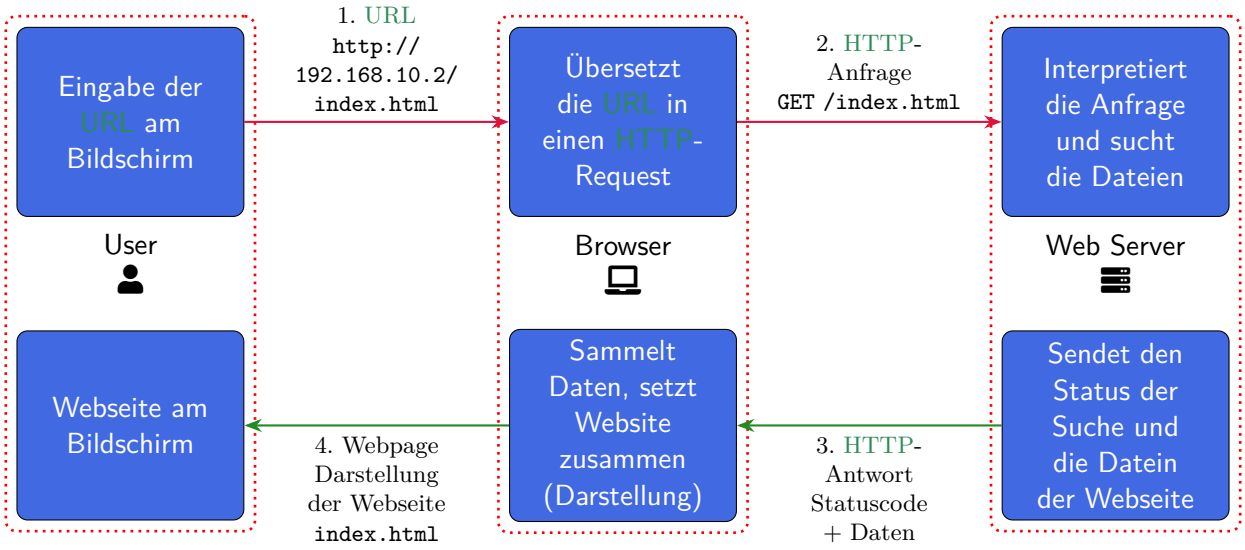


Abbildung 1.18: Schematischer Ablauf eines Webseiten-Aufrufs mit dem HTTP-Protokoll

Damit der User die Nachricht jedoch tatsächlich erhält, müssen noch einige weitere Schritte erfolgen. Wie werden beispielsweise die Daten vom Server zum Client transportiert? Dafür müssen wir uns näher mit der Transportschicht auseinandersetzen.

Aufgabe 1.35 Filius-Anwendung: Web-Server

Erstellen Sie in Filius eine Client-Server-Architektur, ähnlich derjenigen aus Video 5. Diesmal soll der Server jedoch als Web-Server fungieren, der das HTTP-Protokoll unterstützt. Installieren Sie dazu auf dem Server einen „Webserver“ und auf dem Client einen „Webbrowser“. Die Dateien, welche dem Client dargestellt werden sollen, befinden sich bereits auf dem Server. Installieren Sie auf dem Server zusätzlich die Software „Text-Editor“ und öffnen Sie die Datei `index.html`, im Ordner `webserver`, um den Inhalt der Webseite anzupassen. Fügen Sie einen neuen Titel und Text hinzu, der auf der Webseite angezeigt werden soll, und testen Sie von einem anderen Client aus, ob die Änderungen korrekt angezeigt werden.

Aufgabe (Challenge) 1.36 Filius-Anwendung: “Echter” Webserver via Modem (zu zweit)

In Filius können Sie auch einen “echten” Webserver einrichten, der über ein Modem den anderen Rechnern im gleichen LAN zur Verfügung steht.

Diese Aufgabe funktioniert nur, wenn Sie sich im gleichen Subnetz befinden. Sie sollten daher einen persönlichen Hotspot (z.B. via Mobiltelefon) erstellen, mit welchem sich beide Personen verbinden können.

Richten Sie nun in Filius einen Webserver ein, der über ein Modem mit dem LAN verbunden ist. Das Modem muss ihre lokale IP-Adresse haben. Belassen Sie den Standard-Port 12345. Richten Sie anschliessend auf einem anderen Rechner im gleichen LAN auf Filius ein Netzwerk mit einem Client und einem Modem ein, das mit dem gleichen LAN verbunden ist. Verwenden Sie als IP-Adresse im Modem die lokale IP-Adresse der ersten Person, bei welcher der Webserver läuft.

Versuchen Sie zunächst, den Webserver der ersten Person von der zweiten Person aus mit ei-

nem `> ping`-Befehl zu erreichen. Öffnen Sie anschliessend den Webbrowser auf dem Client der zweiten Person und geben Sie die IP-Adresse des Webservers der ersten Person ein, um auf die Webseiten der ersten Person zuzugreifen.

Bemerkung 1.4 (Modem vs. Router)**:**
In Filius gibt es sowohl Modems als auch Router. Ein **Modem** (**Mo**dulator-**Dem**odulator) ist ein Gerät, das digitale Signale in analoge Signale umwandelt und umgekehrt, um die Kommunikation über Telefonleitungen oder Kabelverbindungen zu ermöglichen. Ein **Router** hingegen ist ein Gerät, das Datenpakete zwischen verschiedenen Netzwerken weiterleitet und den Datenverkehr innerhalb eines Netzwerks verwaltet. In Filius wird ein Modem verwendet, um eine Verbindung zu einem externen Netzwerk herzustellen, während ein Router verwendet wird, um den Datenverkehr innerhalb eines lokalen Netzwerks zu steuern. In Wirklichkeit sind viele moderne Geräte eine Kombination aus Modem und Router, die beide Funktionen in einem einzigen Gerät vereinen.

Aufgabe (Challenge) 1.37 HTML genauer verstehen

Besuchen Sie folgende Webseite: `https://www.w3schools.com/html` und lesen Sie die Grundlagen von HTML. Erstellen Sie anschliessend eine einfache HTML-Datei mit einem Titel, einer Überschrift und einem Absatz. Speichern Sie die Datei als `meine_erste_webseite.html` und öffnen Sie sie in Ihrem Webbrowser, um das Ergebnis zu sehen.

1.6.4 DNS-Server

Wie wir bereits gesehen haben, besteht eine URL, also eine Adresse einer Webseiten, aus der Angabe eines Servers und einem Dateipfad. Diese Datei fordert der Client von einem Server mittels dem HTTP- oder Hypertext Transfer Protocol Secure (HTTPS)-Protokoll an (siehe Abbildung 1.18). Allerdings möchte man sich nicht für jeden Server die IP-Adresse merken müssen: ähnlich Ihrer Kontakteliste auf dem Mobiltelefon, welche dazu dient, dass Sie sich nicht alle Telefonnummern auswending merken müssen, ist eine DNS essentiell eine Liste von merkbaren Webseiten-Namen (wie beispielsweise `sbb.ch`) und deren dazugehörigen IP-Adressen, also den IP-Adressen der Server, die die Webseiten-Datein speichern. Ein Beispiel einer solchen Tabelle ist in Tabelle 1.8 gezeigt.

Domain Name	IP-Addresse
`beispiel.ch`	192.168.0.11
`schule.ch`	192.168.0.12
`test.org`	192.168.0.13
`insta.com`	192.168.0.14
...	...

Tabelle 1.8: Beispiel einer DNS-Tabelle

Wenn eine Adresse wie `insta.org` im Browser eingetippt und angefordert wird, schickt der Client eine Anfrage an einen nahegelegenen DNS-Server, der die IP-Adresse retourniert (ähnlich einem Telefonbuch). Ein DNS ist also ein Server, welcher den Clients eines Netzwerks eine Tabelle von

Servern und damit verbundenen Namen zur Verfügung stellt. Eine solche DNS-Abfrage ist illustriert in Abbildung 1.19.

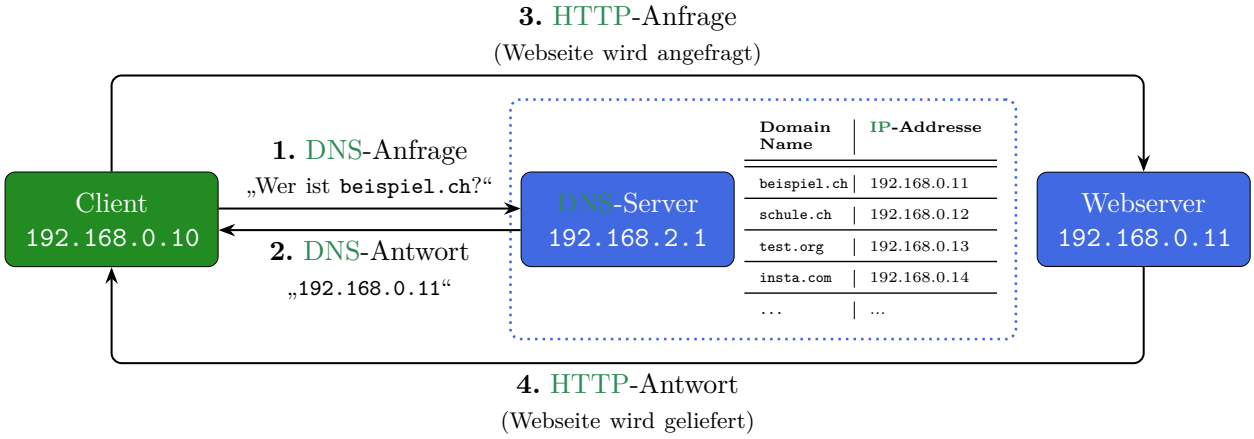


Abbildung 1.19: Schematischer Ablauf einer DNS-Anfrage

Der Prozess der Beantwortung einer Frage nach der IP-Adresse einer Web-Adresse ist in Schritten 1 und 2 in Abbildung 1.19 gezeigt. Die Übersetzung eines Domain-Namens in eine IP-Adresse wird auch als **Namensauflösung** bezeichnet.

Aufgabe 1.38 DNS-Server: Verständnisfrage

Weshalb kann der Client `192.168.0.10` bei Abbildung 1.19 nicht auf die Webseite `ksimlee.ch` zugreifen?

Mit dem Terminal-Befehl `> host [ipadress]` kann nachgeschaut werden, welche IP-Adresse zu einer bestimmten Webseite gehört. Mit dem Befehl `> nslookup [ipadress]` erhalten Sie dieselben Informationen und zusätzlich dazu noch die IP-Adresse des DNS-Servers, der Ihnen die Informationen geliefert hat.

Aufgabe 1.39 IP-Adresse einer Webseite herausfinden

Verwenden Sie die Befehle `> host` oder `> nslookup` in der Befehlszeile, um herauszufinden, welche IP-Adresse zur Webseite `sbb.ch` gehört. Geben Sie diese IP im URL-Fenster Ihres Browser ein und beobachten Sie, was passiert.

Aufgabe 1.40 Filus-Anwendung: DNS-Server

Erstellen Sie in Filius eine Client-Server-Architektur, ähnlich derjenigen aus Video 5, allerdings mit 2 Servern, wovon der erste als DNS-Server und der zweite als Web-Server fungiert. Der DNS-Server soll dem Client die IP-Adresse des Web-Servers bereitstellen, damit der Client anschliessend eine Webseite vom Web-Server anfordern kann. Schauen Sie sich hierzu auf Filius um, welche Software-Pakete für die jeweiligen Server zur Verfügung stehen. Vergessen Sie nicht, den DNS-Server auch bei jedem Client einzutragen, damit dieser weiss, an welchen Server er die DNS-Anfragen schicken muss!

1.7 Weitere Aufgaben

Aufgabe 1.41 Video zu Netzwerkschichten und Protokollen (Zusammenfassung)

Um die Netzwerkschichten und Protokolle visuell zu verstehen, schauen Sie sich folgendes Video an:

Youtube-Video (13')

Notieren Sie sich dabei Ihre Antworten auf folgende Fragen:

- Welche **Schichten**, **Geräte** und **Protokolle** werden genannt?
  - Namen (Abkürzung und ganz)
  - Wofür sind sie zuständig?
  - Welche Protokolle gehören zu welchen Schichten?
  - Welche Geräte gehören zu welchen Protokollen / Schichten?
- Notieren Sie Ihre **Fragen**: Welche Begriffe verstehen Sie (noch) nicht?

Aufgabe 1.42 Austausch zu Netzwerkschichten und Protokollen

Tauschen Sie sich nun in 3er- bis 4er-Gruppen aus und tragen Sie ihre Notizen zusammen. Beantworten Sie danach folgende Fragen:

- Was ist eine URL?
- Was macht „Mr. IP"?
- Was macht der Router?
- Was macht die Firewall?
- Was macht der Proxy Server?
- Was macht der „Ping of Death"? 💀

Aufgabe (Challenge) 1.43 Weitere Filius-Aufgaben (Troubleshooting)

Lösen Sie die Troubleshooting-Aufgaben 2.13 auf folgendem Link.

Aufgabe (Challenge) 1.44 Vertiefung: Schichtenmodell

Lesen Sie zur Vertiefung des Schichtenmodells folgende Webseite (ohne Übungen): Link

Kapitel 2

Lernaufgabe: Netzwerke

Ziel: Am Ende soll ein funktionsfähiges Programm stehen, mit dem die Lernenden in Echtzeit Nachrichten austauschen können.

Problembezug: „Wie kann ich mit einer anderen Person eine Nachricht über das Internet austauschen?“

Komplexe Lernaufgabe: Chat-Anwendung

Die **Haupt-Lernaufgabe** lautet: Entwickeln Sie in Kleingruppen eine **funktionierende Chat-Anwendung** in Python, die über einen Relay-Server läuft. Am Ende sollen die Gruppen miteinander in Echtzeit Nachrichten austauschen können. Diese Haupt-Lernaufgabe ist in mehrere **Learning Tasks** unterteilt, die aufeinander aufbauen und jeweils durch unterstützende Informationen, Just-in-time Hilfen und Part-task Practice (Teilübungen) begleitet werden.

Scaffolding der Lernaufgaben

Die Unterrichtssequenz besteht aus drei aufeinander aufbauenden Learning Tasks, welche jeweils durch vorausgehende, unterstützende Informationen, Just-in-time Hilfen und Part-task Practice (Teilübungen) begleitet werden. Dieses *Scaffolding* jedes Learning Tasks wird jeweils in einer blauen Box hervorgehoben, ebenso wie das fortschreitende Zurückweichen (*Fading*) der Unterstützung.

Unterstützende Informationen	**Just-in-time Informationen**	**Part-task Practice (Teilübungen)**
• Visualisierung der Client-Server-Architektur (Skript, Slides) • Einführung in grundlegende Begriffe: IP-Adresse, Port, Socket • Analogie: „Postsystem“ (Adresse → Empfänger) • Infoblätter / Moodle-Materialien mit Hintergrundwissen	• Code-Snippets für `socket.connect`, `send`, `recv` • Anleitung zur Fehlerbehandlung bei Verbindungsabbrüchen • Debugging-Tipps, wenn Nachrichten nicht ankommen • Schritt-für-Schritt-Hilfekarten (nur bei Bedarf zugänglich)	• **Erste Übung:** Zwei Python-Skripte auf demselben Rechner verbinden • Übung: Eine „Hallo“-Nachricht an den Server senden und Antwort empfangen • Fehlerübungen: Verbindungsfehler identifizieren und beheben • Mini-Tests zu IP/Port-Adressen und deren Bedeutung

Tabelle 2.1: Didaktische Struktur der Lernaufgabe: Chat-Kommunikation

2.1 Learning Task 1: Lokale Verbindung via `localhost`

Vorausgehende Konzepte: Die SuS kennen bereits die historischen Hintergründe von Netzwerken, die Grundkonzepte von Netzwerkschichten, IP-Adressen, Sequenzierung und Ports, sowie die Client-Server-Architektur aufgrund einer vorangehenden Unterrichtssequenz. In der Vorausgehenden Unterrichtssequenz haben sich die SuS mit den grundlegenden Funktionen und der Bedeutung von Netzwerken auseinandergesetzt, indem Sie ein Spiel gespielt haben, bei dem sie Nachrichten über ein simuliertes Netzwerk via ausgedruckte Kärtchen zusenden mussten (s. Kapitel A). Dabei haben sie die Rolle von IP-Adressen und Ports kennengelernt und verstanden, wie Datenpakete in einem Netzwerk weitergeleitet werden.

Neue Konzepte: Client-Server-Architektur, IP-Adresse, Port, Socket, localhost.

Die neuen Begriffe werden zuerst anhand eines Videos (Warriors of the Net) und einer Analogie (Postsystem) eingeführt. Anschliessend wird die Grundstruktur eines Socket-Programms in Python vorgestellt. Die SuS lösen dabei einige vorbereitete Teilübungen (Part-task Practice), um die Konzepte zu verinnerlichen, und lösen danach die komplexe Lernaufgabe, bei der eine lokale Chat-Kommunikation über zwei Python-Skripte realisiert wird.

Aufgabe 2.1 Chatten auf demselben Rechner

Aufgabe: Schreiben Sie zwei Python-Programme, welche auf demselben Rechner laufen: `server_localhost.py` und `client_localhost.py`. Der Client verbindet sich mit dem Server und schickt eine kurze Nachricht: „Hallo Server, hier ist der Client“. Der Server empfängt die Nachricht und gibt sie auf der Konsole aus, zudem antwortet er an den Client „Hallo Client, hier ist der Server“.

Verwenden Sie folgende Grundstruktur:

```
import socket

PORT = ... # HIER Port-Nummer Ihrer Wahl einsetzen
```

```python
# Server-Socket erstellen
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(("localhost", PORT))   # lokale IP, Port 12345
server.listen(1)
print("Warte auf Verbindung...")

# Verbindung akzeptieren
conn, addr = server.accept()
print(f"Verbunden mit {addr[0]} über Port {addr[1]}")

# Nachricht empfangen
nachricht = conn.recv(1024).decode()
print(f"Empfangen: {nachricht}")

# Antwort an den Client senden
conn.send("Hallo Client, hier ist der Server!".encode())

conn.close()
server.close()
```

Programm 2.1: server_localhost.py

```python
import socket

PORT = ... # HIER Port-Nummer Ihrer Wahl einsetzen

# Client-Socket erstellen und mit dem Server verbinden
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(("localhost", 12345))

# Nachricht an den Server senden
client.send("Hallo Server, hier ist der Client!".encode())

# Antwort vom Server empfangen
antwort = client.recv(1024).decode()
print("Antwort vom Server:", antwort)

client.close()
```

Programm 2.2: client_localhost.py

`socket.AF_INET` bedeutet, dass IPv4-Adressen verwendet werden, und `socket.SOCK_STREAM` steht für eine TCP-Verbindung. Andere, gängige Optionen sind `socket.SOCK_DGRAM` für UDP-Verbindungen und `socket.AF_INET6` für IPv6-Adressen.

Wählen Sie einen geeigneten Port (zwischen 1024 und 65535) und ersetzen Sie die Platzhalter in beiden Programmen.

Lassen Sie `server_localhost.py` auf der Konsole laufen und führen Sie dann `client_localhost.py` aus, indem Sie in VS Code auf den Dropdown-Button rechts vom Run-Button clicken und „Run Python File in Dedicated Terminal“ anklicken.

Erweitern Sie nun Ihr Programm, indem der Client eine Nachricht schickt, und der Server antwortet mit „Nachricht erhalten!“.

Achtung

Wichtiger Hinweis 2.1 (Gebundene Ports)**:**
Wenn Sie den Server beenden, wird der Port unter Umständen für kurze Zeit weiterhin als „belegt“ angezeigt. Dies liegt daran, dass das Betriebssystem den Port für eine gewisse Zeit reserviert, um sicherzustellen, dass alle Daten korrekt übertragen wurden. Wenn Sie versuchen, den Server sofort neu zu starten und denselben Port zu verwenden, kann es zu einem Fehler kommen, da der Port noch als belegt gilt. In diesem Fall können Sie folgende Zeile hinzufügen, um den Port sofort wiederverwendbar zu machen:

```
server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
```

Fügen Sie diese Zeile direkt nach der Erstellung des Server-Sockets ein, also nach folgender Zeile:

```
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Aufgabe 2.2 Verständnisfragen

- Was geschieht, wenn in `server_localhost.py` die IP-Adresse `"localhost"` durch `"127.0.0.1"` ersetzt wird?
- Was geschieht, wenn eine andere Port-Nummer als im Client und Server verwendet wird? Funktioniert die Verbindung? Weshalb (nicht)?
- Warum muss der Server gestartet werden, bevor der Client versucht, sich zu verbinden?
- Was passiert, wenn der Server eine Nachricht sendet, aber der Client keine Antwort erwartet?

Aufgabe 2.3 Reflexionsfragen (formatives Assessment)

Notiere nach Abschluss von Learning Task 1 (max. 5 Minuten) kurze Antworten ins Lernjournal. Tausche dich danach mit einer Person aus und ergänze fehlende Punkte.

Fragen:

1. Erkläre in *eigenen Worten*: Was ist ein Socket? Welche zwei Operationen hast du verwendet?
2. Unterschied Socket vs. Port?
3. Unterschied IP-Adresse vs. Port (verwende die Post-Analogie).
4. Warum muss der Server vor dem Client gestartet werden? Welche konkrete Fehlermeldung erscheint sonst?
5. Weshalb muss `bind()` nur einmal aufgerufen werden?
6. Skizziere den Ablauf deiner Nachricht (Methodenaufrufe in Reihenfolge) vom Client zum Server.
7. Nenne mindestens einen Debugging-Schritt, falls keine Antwort zurückkommt.
8. Formuliere eine mögliche Ursache für `ConnectionRefusedError`.

9. Kurzer Selbstcheck (Ja/Nein):
   - Ich kann den Code ohne Vorlage erneut schreiben.
   - Ich verstehe jede Zeile von `server.py`.
   - Ich weiss, wann `recv` blockiert.
10. Formulieren Sie ein persönliches Lernziel für Task 2 (ein Satz).

2.2 Learning Task 2: Chat-Kommunikation über ein lokales Netzwerk

Neue Konzepte: IP-Adresse im lokalen Netzwerk, Firewall-Einstellungen.

Die SuS lernen, wie sie die eigene IP-Adresse im lokalen Netzwerk ermitteln können (z. B. über den Befehl `ipconfig` in der Windows-Eingabeaufforderung oder `ifconfig` im Terminal auf macOS/Linux). Zudem wird erklärt, wie Firewall-Einstellungen den Datenverkehr beeinflussen können. Die Teilübungen (Part-task Practice) helfen den SuS, ihre Programme anzupassen und Verbindungsprobleme zu lösen. Am Ende steht die komplexe Lernaufgabe, bei der die Chat-Kommunikation über zwei Geräte im selben Netzwerk realisiert wird.

Technische Herausforderungen: Die Lehrperson sollte vorab testen, ob die Firewall-Einstellungen auf den Schulrechnern die Kommunikation über Sockets erlauben. Gegebenenfalls müssen Ausnahmen für die verwendeten Ports eingerichtet werden oder ein eigenes Netzwerk (z. B. über einen mobilen Hotspot oder einen eigenen WLAN-fähigen Router) genutzt werden.

Aufgabe 2.4 Meine Öffentliche IP-Adresse herausfinden

Finden Sie die lokale IP-Adresse Ihres Laptops heraus. Dies ist die IP, welche Sie innerhalb des lokalen Netzwerks (z.B. Schul-WLAN) verwenden. Verwenden Sie dazu die Anleitungen in Aufgabe 1.12.

Ihre private IP-Adresse sollte in etwa so aussehen: `192.168.x.x` oder `10.x.x.x`.

Meine private IP-Adresse ist: . . .

Notieren Sie sich auch noch ihre öffentliche IP-Adresse, welche Sie im Internet verwenden. Diese finden Sie z.B. auf der Webseite `https://www.whatismyip.com/`.

Meine öffentliche IP-Adresse ist: . . .

Aufgabe 2.5 Chatten im lokalen Netzwerk

Aufgabe: Erweitern Sie Ihre Codes aus Aufgabe 2.1, so dass ihre Programme auf **verschiedenen Rechnern im selben lokalen Netzwerk** laufen. Der Server lauscht auf seiner eigenen, lokalen IP-Adresse und einem Port (z.B. 12345). Der Client verbindet sich mit der IP-Adresse des Servers und schickt eine kurze Nachricht („Hallo Server“). Der Server empfängt die Nachricht und gibt sie auf der Konsole aus.

Schritte:

1. Ermitteln Sie die lokale IP-Adresse ihres Rechners und desjenigen ihrer Pultnachbarin (s. Aufgabe 1.12). Bestimmen Sie, wer von Ihnen die Rolle des Servers und wer die Rolle des Clients übernimmt.
2. Passen Sie die `bind`- und `connect`-Befehle in beiden Skripts entsprechend an.
3. Starten Sie `server.py` auf Rechner A und `client.py` auf Rechner B.

Aufgabe 2.6 Endlos-Chat

Im echten Leben wollen wir nicht nur eine einzige Nachricht austauschen, sondern mehrere Nachrichten hin- und herschicken. Erweitern Sie Ihre Programme so, dass der Client und der Server in einer Endlosschleife Nachrichten austauschen können, bis einer der beiden die Verbindung mit `Strg + C` abbricht. Verwenden Sie dafür folgende Grundstruktur:

```
try:
    while True:
        # Nachricht senden / empfangen
except KeyboardInterrupt:
    print("Verbindung wird geschlossen.")
    # Socket schliessen
```

Aufgabe 2.7 Reflexionsfragen (formatives Assessment)

Notiere nach Abschluss von Learning Task 2 (max. 5 Minuten) kurze Antworten ins Lernjournal. Tausche dich danach mit einer Person aus und ergänze fehlende Punkte.

Fragen:

1. Wie unterscheidet sich die IP-Adresse im lokalen Netzwerk von der `localhost`-Adresse?
2. Warum ist es wichtig, die richtige IP-Adresse des Servers zu verwenden?
3. Welche Fehlermeldung erscheint, wenn der Client versucht, sich mit einer falschen IP-Adresse zu verbinden?
4. Nenne mindestens einen Debugging-Schritt, falls keine Verbindung hergestellt werden kann.
5. Wie kannst du überprüfen, ob die Firewall die Verbindung blockiert?
6. Kurzer Selbstcheck (Ja/Nein):
   - Ich verstehe jede Zeile von `server.py`.
   - Ich weiss, wie ich die lokale IP-Adresse meines Rechners finde.
7. Ein persönliches Lernziel für Task 3 (ein Satz).
8. Welche Herausforderungen könnten auftreten, wenn Sie versuchen, über das Internet zu kommunizieren, anstatt nur im lokalen Netzwerk?

2.3 Learning Task 3: Chat-Kommunikation über das LAN, mittels Relay-Server

Neue Konzepte: Relay-Server, Client-Identifikation.

Die SuS lernen das Konzept eines Relay-Servers kennen, der als Vermittler zwischen mehreren Clients fungiert. Der Relay-Server empfängt Nachrichten von einem Client und leitet sie an den entsprechenden Empfänger-Client weiter. Die Teilübungen (Part-task Practice) helfen den SuS, ihre Programme anzupassen und die Logik für die Client-Identifikation zu implementieren. Am Ende steht die komplexe Lernaufgabe, bei der die Chat-Kommunikation über einen Relay-Server im lokalen Netzwerk realisiert wird.

2.3.1 Überblick: Wie funktioniert ein Chat-Client?

Ein Chat-Client verbindet sich über das Internet mit einem Server, also einem Computer, der Nachrichten von Clients hin- und herschickt. Nachrichten werden als Text über ein das TCP-Protokoll verschickt. Der Server kann die Nachricht an andere Clients weiterleiten oder – wie beim Echo-Server – einfach zurückschicken.

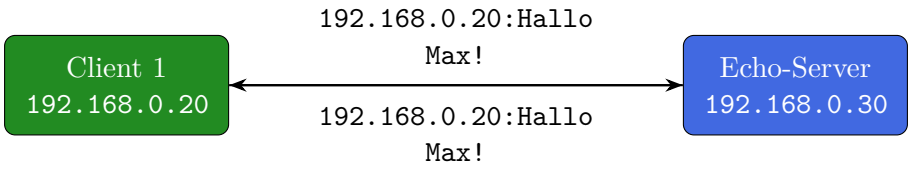


Abbildung 2.1: Ablauf einer Chat-Nachricht über den Echo-Server

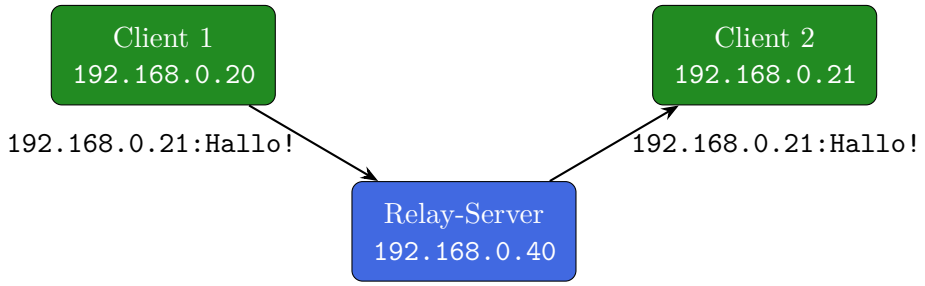


Abbildung 2.2: Kommunikation über einen Relay-Server: Die Nachricht wird vom Client an den Relay-Server geschickt und von dort an den Ziel-Client weitergeleitet.

Im folgenden werden wir die Kommunikation über einen Relay-Server umsetzen. So können Sie Nachrichten an MitschülerInnen schicken, ohne deren IP-Adresse zu kennen.

2.3.2 Aufgaben: Schritt für Schritt zum Chat-Client

Aufgabe 2.8 Nachricht zusammensetzen

Schreiben Sie ein Python-Programm, das folgende Variablen definiert:

- `ip`: die lokale IP-Adresse Ihres Mitschülers/Ihrer Mitschülerin als Text
- `message`: die Nachricht, die Sie schicken möchten
- `combined`: die kombinierte Chat-Nachricht im Format `ip:message`

Geben Sie den Wert von `combined` mit `print()` aus.

Aufgabe 2.9 Benutzereingabe und Schleife

Erweitern Sie Ihr Programm so, dass die IP-Adresse und die Nachricht jeweils per `input()` eingegeben werden. Kombinieren Sie beide wie oben und geben Sie das Ergebnis aus. Bauen Sie eine Endlosschleife ein, damit Sie beliebig viele Nachrichten verschicken können.

Aufgabe 2.10 Server-IP und Port definieren

Definieren Sie die IP-Adresse und den Port des Chat-Servers als Variablen `server_ip` und `port` (siehe Tabelle unten).

Aufgabe 2.11 Verbindung zum Server herstellen

Das Paket `socket` ist in Python standardmässig bereits installiert und dient dazu, um Netzwerkverbindungen herzustellen. Importieren Sie das Paket und erstellen Sie eine Funktion `sende`, die sich mit dem Server verbindet und eine Nachricht sendet. Die Funktion soll die IP-Adresse des Servers, den Port und die Nachricht als Parameter entgegennehmen. Die Antwort des Servers soll ausgegeben werden.

Verwenden Sie folgende Code-Vorlage: Innerhalb Ihrer Definition:

```
import socket

# IHR BISHERIGER CODE HIER

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s: # Socket
    erstellen und verbinden
    s.connect((server_ip, port)) # Mit Server verbinden
    s.sendall(message.encode()) # Nachricht senden
    data = s.recv(1024) # Antwort empfangen und ausgeben
    print('Antwort vom Server:', data.decode())
```

Aufgabe 2.12 Chat-Client zusammenbauen und testen

Kombinieren Sie alles zu einem vollständigen Chat-Client, so dass Sie in einer Endlosschleife Nachrichten eingeben und an den Server schicken können. Die Antwort des Servers wird ausgegeben.

Der Client schickt eine Nachricht im Format `IP:Nachricht` an den Server. Der Echo-Server schickt

exakt dieselbe Nachricht zurück. So können Sie testen, ob Ihr Client korrekt funktioniert.

2.4 Learning Task 4: Chat-Kommunikation über das Internet, mittels Relay-Server

Bis jetzt haben wir nur wenige Nachrichten innerhalb des lokalen Netzwerks ausgetauscht. In der Praxis möchten wir jedoch oft mit Personen kommunizieren, die sich an einem anderen Ort befinden, also über das Internet. Zudem möchten wir einen Endlos-Fluss an Nachrichten senden und empfangen können, ohne dass die Verbindung nach jeder Nachricht getrennt wird. In diesem Kapitel lernen wir, wie dieses mit einem Relay-Server im lokalen Netzwerk sowie Python realisiert werden kann.

Neue Konzepte: Öffentliche IP-Adresse, NAT, Port-Forwarding, Relay-Server, DNS (Domain Name System).

Die SuS lernen, wie sie die öffentliche IP-Adresse ihres Netzwerks ermitteln können (z. B. über Websites wie whatismyipaddress.com). Zudem wird das Konzept von NAT (Network Address Translation) und Port-Forwarding erklärt, um zu verstehen, warum direkte Verbindungen oft nicht möglich sind. Zusätzlich wird das Domain Name System (DNS) eingeführt: DNS übersetzt menschenlesbare Domain-Namen (wie `chat.example.com`) in IP-Adressen, damit Programme Server im Internet finden können. Die Teilübungen (Part-task Practice) helfen den SuS, ihre Programme anzupassen und Verbindungsprobleme zu lösen. Am Ende steht die komplexe Lernaufgabe, bei der die Chat-Kommunikation über einen Relay-Server realisiert wird.

Technische Herausforderungen: Die Lehrperson sollte vorab testen, ob die Firewall-Einstellungen auf den Schulrechnern die Kommunikation über Sockets erlauben. Gegebenenfalls müssen Ausnahmen für die verwendeten Ports eingerichtet werden oder ein eigenes Netzwerk (z. B. über einen mobilen Hotspot oder einen eigenen WLAN-fähigen Router) genutzt werden.

Aufgabe 2.13 Chatten über einen öffentlichen Relay-Server

Aufgabe: Sie verbinden sich mit einem bereits laufenden öffentlichen Relay-Server, der Nachrichten zwischen mehreren Clients weiterleitet. Ziel ist, dass Sie und Ihre Mitschüler:innen sich gegenseitig Nachrichten schicken können.

Schritte:

1. Öffnen Sie `client.py` und passen Sie die `connect`-Zeile an:

   ```
   client.connect(("88.198.193.239", 12345))
   ```
2. Senden Sie eine Nachricht (z.B. „Hallo an alle!“) an den Server.
3. Empfangen Sie die Antwort und geben Sie sie auf der Konsole aus.
4. Starten Sie `client.py` auf mehreren Rechnern gleichzeitig und tauschen Sie Nachrichten aus.
5. Probieren Sie aus, ob Sie von verschiedenen Rechnern aus gleichzeitig Nachrichten senden und empfangen können. Was passiert, wenn mehrere Clients verbunden sind? Können Sie sich gegenseitig Nachrichten schicken?
6. **Erweiterung:** Jeder Client gibt beim Start einen eigenen Namen ein. Beim Senden einer Nachricht wählen Sie aus, an welchen Namen (also an welchen anderen Client)

die Nachricht geschickt werden soll. Die Nachricht wird dann nur an diesen Empfänger weitergeleitet.

Verwenden Sie folgende Grundstruktur:

```
import socket

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(("88.198.193.239", 12345))  # IP-Adresse des Relay-Servers,
    Port 12345

name = input("Dein Name: ")
client.send(name.encode())  # Namen an Server senden

def empfange():
    while True:
    antwort = client.recv(1024).decode()
    if not antwort:
        break
    print(antwort)

import threading
threading.Thread(target=empfange, daemon=True).start()

while True:
    empfaenger = input("Empfänger-Name (oder 'exit'): ")
    if empfaenger.lower() == "exit":
    break
    nachricht = input("Nachricht: ")
    if nachricht.lower() == "exit":
    break
    # Format: EMPFAENGER:Nachricht
    client.send(f"{empfaenger}:{nachricht}".encode())

client.close()
```

Programm 2.9: client.py

Hinweis: Der Server muss die Namen der verbundenen Clients verwalten und Nachrichten entsprechend weiterleiten. So können Sie gezielt Nachrichten an einzelne Personen schicken.

Aufgabe 2.14 Reflexionsfragen (formatives Assessment)

Notiere nach Abschluss von Learning Task 3 (max. 5 Minuten) kurze Antworten ins Lernjournal. Tausche dich danach mit einer Person aus und ergänze fehlende Punkte. **Fragen:**

1. Was ist der Unterschied zwischen einer lokalen IP-Adresse und einer öffentlichen IP-Adresse?
2. Erkläre kurz, was NAT (Network Address Translation) ist und warum es in Heimnetzwerken verwendet wird.
3. Warum ist Port-Forwarding notwendig, um Verbindungen von aussen zu einem Gerät im lokalen Netzwerk herzustellen?
4. Was ist ein Relay-Server und welche Rolle spielt er in diesem Szenario?
5. Wie funktioniert DNS (Domain Name System) und warum ist es wichtig für die Kommunikation im Internet?
6. Kurzer Selbstcheck (Ja/Nein):
   - Ich kann den Code ohne Vorlage erneut schreiben.
   - Ich verstehe jede Zeile von `client.py`.
   - Ich weiss, wie ich die öffentliche IP-Adresse meines Netzwerks finde.
7. Reflektiere: Welche Herausforderungen könnten auftreten, wenn Sie versuchen, über das Internet zu kommunizieren, anstatt nur im lokalen Netzwerk?

Anhang A

Netzwerk-Spiel

In dieser Übung sollen Sie die verschiedenen Schichten des Netzwerk-Schichtenmodells anhand eines praktischen Beispiels anwenden. Dabei schicken Sie sich gegenseitig Nachrichten zu, indem sie die IP-Adresse der anderen Person verwenden. Die IP-Adresse wurde im Vorfeld von der Lehrperson zugeteilt, welche ebenfalls die Rolle des Routers (innerhalb eines lokalen Netzwerks) übernimmt. Das Spiel läuft wie folgt ab:

1. Jede Person erhält ein ausgedrucktes Namensschild mit dazugehöriger, zufällig generierter MAC-Adresse, sowie einer IP-Adresse. Das Namensschild wird so platziert, dass es für alle gut sichtbar ist.
2. Die Lehrperson erklärt die verschiedenen Schichten des Netzwerk-Schichtenmodells und wie sie in diesem Spiel angewendet werden.
3. Jede Person schreibt eine kurze Nachricht (z.B. „Hallo, wie geht's?“) und adressiert sie an eine andere Person, indem sie deren IP-Adresse verwendet. Die maximale Nachrichtenlänge beträgt 10 Zeichen.
4. Die Nachricht wird mit einer Sequenznummer versehen, um die Reihenfolge der Nachrichten zu kennzeichnen (Couvert Grösse `A6`).
5. Die Nachricht wird nun mit der IP-Adresse des Empfängers sowie der eigenen IP-Adresse versehen und dem Router übergeben.
6. Der Router kennt nur die MAC-Adressen der Teilnehmer und muss die IP-Adressen in MAC-Adressen auflösen, um die Nachricht korrekt weiterzuleiten. Er tut dies, indem er alle laut fragt, wer eine bestimmte IP-Adresse hat und konsultiert dabei seine ARP-Tabelle. Falls eine Nachricht nicht den Regeln entspricht (z.B. zu lange Nachricht, falsche IP-Adresse), wird sie verworfen und nicht weitergeleitet. Ansonsten leitet der Router die Nachricht an den entsprechenden Empfänger weiter.
7. Die Empfänger öffnen die Couvert-Nachrichten und sortieren die Nachrichten nach Sequenz-Nummer. Sie lesen die Nachrichten laut vor und beantworten sie gegebenenfalls.
8. Die Lehrperson fasst die Übung zusammen und erklärt, wie das Netzwerk-Schichtenmodell in der Praxis funktioniert. Dabei wird auf die verschiedenen Schichten eingegangen und erläutert, wie sie in diesem Spiel umgesetzt wurden.
9. Optional: Die Lehrperson kann die Übung erweitern, indem sie verschiedene Netzwerk-Szenarien simuliert, z.B. Netzwerk-Ausfälle, Verzögerungen oder Störungen, um die Auswirkungen auf die Kommunikation zu verdeutlichen.
10. Mögliche Erweiterung des Spiels: zweiter und dritter Router (Personen) bestimmen, um die Konzepte von Subnetzen und Routing zu verdeutlichen. Konzepte: Netzmaske, Gateway, Interface, Routing-Tabelle.

③ Von:	
③ An:	
② Sequenz:	
① Nachricht (max. 10 Zeichen):	

③ Von:	
③ An:	
② Sequenz:	
① Nachricht (max. 10 Zeichen):	

③ Von:	
③ An:	
② Sequenz:	
① Nachricht (max. 10 Zeichen):	

③ Von:	
③ An:	
② Sequenz:	
① Nachricht (max. 10 Zeichen):	

③ Von:	
③ An:	
② Sequenz:	
① Nachricht (max. 10 Zeichen):	

③ Von:	
③ An:	
② Sequenz:	
① Nachricht (max. 10 Zeichen):	

③ Von:	
③ An:	
② Sequenz:	
① Nachricht (max. 10 Zeichen):	

③ Von:	
③ An:	
② Sequenz:	
① Nachricht (max. 10 Zeichen):	

③ Von:	
③ An:	
② Sequenz:	
① Nachricht (max. 10 Zeichen):	

③ Von:	
③ An:	
② Sequenz:	
① Nachricht (max. 10 Zeichen):	

③ Von:	
③ An:	
② Sequenz:	
① Nachricht (max. 10 Zeichen):	

③ Von:	
③ An:	
② Sequenz:	
① Nachricht (max. 10 Zeichen):	

Anhang B

Installation Filius

B.1 Windows

Laden Sie folgende Datei herunter und installieren Sie das Programm: Link.

Falls Sie dazu aufgefordert werden, klicken Sie immer auf „erlauben“, bzw. „weiter“. Nachdem Sie das Programm installiert haben, können Sie folgende Schritte ausführen:

1. Das heruntergeladene Installationsprogramm (mit der Dateiendung `.exe`) löschen
2. Auf der Tastatur tippen: ⊞+`Filius`
3. Filius öffnen

B.2 MacOS

Achtung

Falls Probleme beim Ausführen der unten stehenden Schritte auftreten, schauen Sie sich die genauen Anleitungen mit Screenshots auf dieser Webseite an. Sie können natürlich auch mich um Hilfe fragen 😉.

Installation Java SDK (Voraussetzung für Filius) Damit Filius ausgeführt werden kann, muss zuerst Java SDK in der neusten Version installiert werden.

1. Laden Sie Java als DMG-Installer unter folgendem Link herunter: `https://www.oracle.com/java/technologies/downloads/`
   - Überprüfen Sie, ob Sie über einen Mac mit Apple- oder Intel-Chip verfügen, indem Sie ganz oben links auf Ihrem Bildschirm auf das -Zeichen → „Über diesen Mac“ klicken und die Beschreibung Ihres Chips (erste Zeile) lesen.
   - Falls Sie einen neueren Mac mit Apple-Silicon-Chip (z.B. M1, M2, M3...) haben (Modelle nach 2022 gekauft), laden Sie die „ARM“-Version herunter (als `.dmg`-Datei)
   - Falls Sie einen Mac mit Intel-Chip haben, laden Sie die „x64“-Version herunter

Installieren Sie Java SDK, indem Sie auf die heruntergeladene Datei klicken.

Gegebenenfalls müssen Sie nach der Installation in den Systemeinstellungen unter „Sicherheit“ erlauben, dass die Java-Anwendung ausgeführt werden darf (unten rechts auf „Trotzdem öffnen“ klicken).

Installation Filius

- Laden Sie Filius in der neusten Version **als `.zip`-Datei** von folgendem Link herunter: https://www.lernsoftware-filius.de/Herunterladen.
- Verschieben (ziehen) Sie den Ordner in Ihren Anwendungs-Ordner.
- Öffnen Sie den Ordner und starten Sie die Datei `filius.jar`, indem Sie rechts darauf klicken (⌥ + Klick) und „Öffnen“ wählen. Dies müssen Sie nur beim ersten Mal machen, danach können Sie „normal“ auf das Programm klicken (ohne ⌥), um es zu öffnen.

Anhang C

NAT und Port Forwarding

C.1 Network Address Translation (NAT)

NAT (Network Address Translation) ist ein Verfahren, das in Routern verwendet wird, um die Kommunikation zwischen Computern in einem lokalen Netzwerk und dem Internet zu ermöglichen. Dabei wird die lokale IP-Adresse eines Computers in eine globale IP-Adresse übersetzt, wenn Datenpakete das lokale Netzwerk verlassen, und umgekehrt, wenn Datenpakete aus dem Internet empfangen werden. Dies ermöglicht es mehreren Computern in einem lokalen Netzwerk, dieselbe globale IP-Adresse zu teilen, wodurch die Anzahl der benötigten globalen Adressen reduziert wird.

Beispiel C.1 (Network Address Translation (NAT))**:**
Stellen Sie sich vor, Sie haben ein lokales Netzwerk mit mehreren Computern, die alle die gleiche globale IP-Adresse `203.0.113.5` teilen. Ein Computer im lokalen Netzwerk mit der lokalen Adresse `192.168.1.10` möchte eine Anfrage an einen externen Server mit der Adresse `8.8.8.8` senden. Der Router übersetzt diese Anfrage wie folgt:

Phase	Quell-IP	Ziel-IP	Ort
Vor NAT	`192.168.1.10` (lokal)	`8.8.8.8`	Lokales Netzwerk
Nach NAT	`203.0.113.5` (global)	`8.8.8.8`	Internet
Rückweg	`8.8.8.8`	`203.0.113.5` (global)	Internet
Nach Rückübersetzung	`8.8.8.8`	`192.168.1.10` (lokal)	Lokales Netzwerk

Tabelle C.1: Beispiel einer NAT-Übersetzung: Ein lokales Gerät kommuniziert mit einem externen Server

Dadurch können mehrere Geräte im lokalen Netzwerk gleichzeitig mit verschiedenen (oder sogar denselben) externen Servern kommunizieren, obwohl sie alle die gleiche globale IP-Adresse nutzen (und potentiell sogar über denselben Port kommunizieren). Der Router kann anhand der durch ihn zugewiesenen Port-Nummern und lokalen Adressen genau unterscheiden, welche Antworten an welche lokalen Geräte zurückgeleitet werden müssen.

NAT wird weiter unterschieden in:

- **Statisches NAT**: Hierbei wird eine feste Zuordnung zwischen jeder lokalen IP-Adresse und einer globalen IP-Adresse erstellt. Dies wird oft für Server verwendet, die von aussen erreichbar

sein müssen. Für private Netzwerke mit vielen Geräten ist dies jedoch unpraktisch, da es viele globale IP-Adressen erfordert.

- **Dynamisches NAT**: Hierbei wird eine lokale IP-Adresse dynamisch einer verfügbaren globalen IP-Adresse zugeordnet, wenn eine Verbindung nach aussen hergestellt wird. Dies ist effizienter als statisches NAT, da nicht für jedes Gerät eine feste globale Adresse benötigt wird. Aufgrund der Tatsache, dass mehrere IP-Adressen benötigt werden, ist dies jedoch immer noch nicht ideal für grosse Netzwerke.
- **Port Address Translation (PAT) (Port Address Translation)**: Auch als **NAT Overloading** bekannt, ermöglicht PAT mehreren Geräten in einem lokalen Netzwerk, dieselbe globale IP-Adresse zu teilen, indem unterschiedliche Port-Nummern verwendet werden. Dies ist die am häufigsten verwendete Form von NAT in Heimnetzwerken, da es die Anzahl der benötigten globalen IP-Adressen minimiert. Jeder lokale Computer wird durch eine Kombination aus der gemeinsamen globalen IP-Adresse und einer eindeutigen Port-Nummer identifiziert In Praxis wird mit NAT indirekt meistens PAT gemeint.

C.2 Port Address Translation (PAT)

Angenommen, Sie haben drei Computer in Ihrem lokalen Netzwerk mit den folgenden lokalen IP-Adressen:

- Computer 1: `192.168.1.10`
- Computer 2: `192.168.1.11`
- Computer 3: `192.168.1.12`

Alle drei Computer teilen sich die gleiche globale IP-Adresse `203.0.113.5`. Wenn alle drei Computer gleichzeitig eine Verbindung zu einem externen Server herstellen möchten, verwendet der Router nicht nur die IP-Adressen zur Unterscheidung, sondern auch die **Port-Nummern**. Der Router erstellt dabei eine **PAT-Tabelle** (häufig auch **NAT-Tabelle** genannt), die nicht nur die lokalen und globalen IP-Adressen, sondern auch die zugehörigen Ports speichert. Die Port-Nummern ermöglichen es dem Router, die eingehenden Antworten vom externen Server korrekt an die jeweiligen lokalen Computer weiterzuleiten, obwohl sie alle dieselbe globale IP-Adresse verwenden.

Dabei ist es sogar möglich, dass mehrere lokale Geräte über denselben lokalen Port kommunizieren, solange die Kombination aus globaler IP-Adresse und globalem Port eindeutig bleibt. Die PAT-Tabelle im Router könnte wie folgt aussehen:

Lokale IP:Port	Globale IP:Port	Ziel-IP:Port	Protokoll	Status
`192.168.1.10:45000`	`203.0.113.5:45000`	`8.8.8.8:443`	TCP	Aktiv
`192.168.1.11:45000`	`203.0.113.5:45001`	`8.8.8.8:443`	TCP	Aktiv
`192.168.1.12:45000`	`203.0.113.5:45002`	`93.184.216.34:80`	TCP	Aktiv

Tabelle C.2: PAT-Tabelle mit mehreren lokalen Geräten, die gleichzeitig auf externe Server zugreifen

In diesem Beispiel verwenden alle drei lokalen Computer zufälligerweise denselben lokalen Port (45000), erhalten jedoch vom Router unterschiedliche globale Ports (45000, 45001, 45002) zugewiesen. Zwei Computer (192.168.1.10 und 192.168.1.11) greifen auf denselben externen Server `8.8.8.8:443` zu, während der dritte Computer (192.168.1.12) auf einen anderen Server `93.184.216.34:80` zugreift. Der Router kann die eingehenden Antworten anhand der Kombination aus globaler IP-Adresse und globalem Port eindeutig dem richtigen lokalen Computer zuordnen.

C.3 Port Forwarding

Port Forwarding kann verwendet werden, um eingehende Verbindungen von externen Computern an bestimmte lokale Computer weiterzuleiten. Dies ist besonders nützlich, wenn ein lokaler Computer (z.B. ein Webserver oder ein Spieleserver) von ausserhalb des lokalen Netzwerks erreichbar sein soll. Durch das Einrichten von Port Forwarding im Router kann der Datenverkehr, der an eine bestimmte Port-Nummer der globalen IP-Adresse gesendet wird, an die entsprechende lokale IP-Adresse und Port-Nummer weitergeleitet werden.

Beispiel C.2 (Port Forwarding)**:**
Angenommen, Sie haben einen Webserver in Ihrem lokalen Netzwerk mit der lokalen IP-Adresse `192.168.1.20` und möchten, dass dieser von ausserhalb Ihres Netzwerks über das Internet erreichbar ist. Ihr Router hat die globale IP-Adresse `203.0.113.5`. Sie richten im Router eine Port-Weiterleitung ein, die eingehende Verbindungen auf Port 80 (HTTP) der globalen Adresse an die lokale Adresse `192.168.1.20` auf Port 80 weiterleitet. Dadurch können externe Benutzer über `http://203.0.113.5` auf Ihren Webserver zugreifen, obwohl dieser sich in einem privaten Netzwerk befindet. Die Forwarding-Tabelle im Router könnte wie folgt aussehen:

Externer Port	**Interne IP**	**Interner Port**	**Protokoll**
80	`192.168.1.20`	80	TCP

Tabelle C.3: Port Forwarding-Tabelle im Router

Das Protokoll kann entweder TCP oder UDP sein, je nachdem, welche Art von Datenverkehr weitergeleitet werden soll. Typischerweise wird TCP für Webserver und andere verbindungsorientierte Dienste verwendet, da es eine zuverlässige Datenübertragung gewährleistet. UDP wird hingegen typischerweise für Anwendungen wie Online-Spiele oder VPN-Dienste genutzt, die eine schnellere, aber weniger zuverlässige Übertragung erfordern.

Beispiel C.3 (NAS im lokalen Netzwerk von aussen erreichbar machen)**:**
Angenommen, Sie kaufen ein Network-Attached Storage (NAS) für Ihr Heimnetzwerk, um Dateien zentral zu speichern und von verschiedenen Geräten darauf zuzugreifen. Ihr NAS hat die lokale IP-Adresse `192.168.1.20`. Sie möchten jedoch auch von ausserhalb Ihres Heimnetzwerks auf das NAS zugreifen können, z.B. wenn Sie unterwegs sind. Ihr Internetanbieter hat Ihnen die globale IP-Adresse `203.0.113.5` zugewiesen. Zudem haben Sie sich eine statische IPv4-Adresse gekauft, damit sich Ihre IP-Adresse nicht ändert.

Um dies zu ermöglichen, müssen Sie Port Forwarding auf Ihrem Router einrichten. Angenommen, Ihr NAS verwendet den Standardport 8080 für den Zugriff auf die Weboberfläche. Sie richten im Router eine Port-Weiterleitung ein, die eingehende Verbindungen auf den Port 8080 der globalen Adresse `203.0.113.5` an die lokale Adresse `192.168.1.20` weiterleitet. Dadurch können Sie von ausserhalb Ihres Heimnetzwerks auf Ihr NAS zugreifen, indem Sie in Ihrem Webbrowser die Adresse `http://203.0.113.5:8080` eingeben.

Die Forwarding-Tabelle im Router könnte wie folgt aussehen:

Externer Port	Interne IP	Interner Port	Protokoll
8080	`192.168.1.20`	8080	TCP

Tabelle C.4: Port Forwarding-Tabelle im Router

Damit Ihr NAS auch wirklich immer die lokale IP-Adresse `192.168.1.20` behält, sollten Sie im Router eine IP-Reservierung für die MAC-Adresse Ihres NAS einrichten (s. Bemerkung 1.3).

Bemerkung C.1 (Unterschied NAT und Port Forwarding)**:**
NAT und Port Forwarding sind eng verwandte Konzepte, die häufig zusammen verwendet werden, aber unterschiedliche Funktionen erfüllen:

- **NAT**: Ist ein automatischer Prozess, der auf dem Router läuft und dafür sorgt, dass Computer im lokalen Netzwerk mit dem Internet kommunizieren können. NAT übersetzt automatisch lokale IP-Adressen in globale Adressen für ausgehende Verbindungen und führt die Rückübersetzung für eingehende Antworten durch. Dies geschieht transparent und erfordert keine manuelle Konfiguration pro Gerät.
- **Port Forwarding**: Ist eine manuell konfigurierte Funktionalität, die es ermöglicht, eingehende Verbindungen von aussen gezielt an spezifische lokale Computer weiterzuleiten. Während NAT primär für ausgehende Verbindungen zuständig ist, ermöglicht Port Forwarding, dass externe Benutzer von aussen initiierte Verbindungen zu Servern im lokalen Netzwerk herstellen können.
- **Zusammenhang**: Port Forwarding funktioniert auf Basis von NAT. Es nutzt die NAT-Tabelle des Routers, um eingehende Verbindungen korrekt an die richtigen lokalen Geräte weiterzuleiten. Ohne NAT wäre Port Forwarding nicht möglich.

Zusammengefasst: NAT ermöglicht Computern im lokalen Netzwerk, nach aussen zu kommunizieren, während Port Forwarding es externen Computern ermöglicht, gezielt auf Dienste im lokalen Netzwerk zuzugreifen.

Anhang D

Lernziele: Rechnernetze

- ☐ Ich kann den Unterschied zwischen Internet und Rechnernetz erklären und Beispiele für verschiedene Netzwerke nennen.
- ☐ Ich kann die Funktionsweise von IP, IPv4 und IPv6 erklären und die Adressräume vergleichen.
- ☐ Ich kann die Erreichbarkeit eines Hosts mit `> ping` prüfen und typische Diagnoseschritte durchführen.
- ☐ Ich kann die Funktionen und Arbeitsweisen von Routern und Switches in einem Netzwerk verstehen und erläutern.
- ☐ Ich kann Subnetzmasken anwenden und einfache Subnetze planen.
- ☐ Ich kann eine Routing-Tabelle interpretieren und für einfache Routing-Entscheidungen nutzen.
- ☐ Ich kann eine Subnetzmaske erstellen und anwenden, um Netzwerke in verschiedene Subnetze zu unterteilen.
- ☐ Ich kann die grundlegenden Konzepte von TCP beschreiben und dessen Einsatz im Kontext von Netzwerkprotokollen verstehen.
- ☐ Ich kann die Rolle von Ports anhand einer Client-Server-Kommunikation erklären.
- ☐ Ich kann Header und Payload eines Datenpakets beschreiben und deren Zweck erklären.
- ☐ Ich kann grundlegende HTML-Strukturen erstellen und verstehen, wie HTML-Seiten über HTTP übertragen werden.
- ☐ Ich kann den Ablauf eines Abrufs von Webinhalten über das Internet in einzelnen Schritten und den darin involvierten Protokollen erläutern.
- ☐ Ich kann die Rolle eines DNS-Servers im Netzwerk erklären und wie DNS-Anfragen bearbeitet und beantwortet werden.
- ☐ Ich kann die grundlegenden Schritte zur Fehlerbehebung einer nicht funktionierenden Verbindung in Filius durchführen und dokumentieren.
- ☐ Ich kann eine lokale Chat-Verbindung über `localhost` aufbauen und testen.
- ☐ Ich kann ein einfaches Socket-Programm in Python analysieren und anpassen (Client und Server).
- ☐ Ich kann eine Chat-Verbindung im LAN einrichten und notwendige Einstellungen (Ports) berücksichtigen.
- ☐ Ich kann die Funktionsweise eines Relay-Servers erklären und einfache Nachrichtenformate definieren.
- ☐ Ich kann mit Netzwerkanalyse-Tools (z.B. Wireshark) gängige Protokolle in einem Abruf identifizieren (DNS, TCP, HTTP).
- ☐ Ich kann Chancen und Risiken zentraler Serverarchitekturen hinsichtlich Datenschutz und Souveränität reflektieren.
- ☐ Ich kann konkrete Massnahmen zur Datenminimierung und zum Schutz der Privatsphäre in einem Chat-Design vorschlagen.

Glossar

AES Advanced Encryption Standard. 13
ARP Address Resolution Protocol. 8, 10–13, 21, 49
ARPANET Advanced Research Projects Agency Network. 2, 3
ASCII American Standard Code for Information Interchange. 8

CIDR Classless Inter-Domain Routing. 24–26

DHCP Dynamic Host Configuration Protocol. 20, 26, 27
DNS Domain Name System. 8, 28, 35, 36, 57

FTP File Transfer Protocol. 8, 28

HTML HyperText Markup Language. 3, 4, 35, 57
HTTP Hypertext Transfer Protocol. 3, 4, 8, 13, 14, 28, 33–36, 55, 57
HTTPS Hypertext Transfer Protocol Secure. 8, 28, 35

ICMP Internet Control Message Protocol. 8, 10, 11
IMAP Internet Message Access Protocol. 28, 32
IoT Internet of Things. 5
IP Internet Protocol. 3, 7–10, 12, 13, 15, 17–27, 33, 35, 36, 41–44, 49, 53–57
ISP Internet Service Provider. 18

LAN Local Area Network. 16, 25–27, 34, 57

MAC Media Access Control. 8–10, 12, 15, 27, 49, 56

NAS Network-Attached Storage. 55, 56
NAT Network Address Translation. 18, 19, 22, 53, 54, 56

OSI Open Systems Interconnection. 8

PAT Port Address Translation. 22, 54
POP3 Post Office Protocol Version 3. 28, 32

SMTP Simple Mail Transfer Protocol. 8, 28, 31, 32
SSH Secure Shell. 28
SSL Secure Sockets Layer. 8, 28

TCP Transmission Control Protocol. 3, 7–9, 27–30, 44, 54–57
TLD Top-Level Domain. 33
TLS Transport Layer Security. 8

UDP User Datagram Protocol. 8, 28, 29, 55

URL Uniform Resource Locator. 3, 4, 33–35
UTF-8 Unicode Transformation Format — 8-bit. 8

VPN Virtual Private Network. 29, 55

WAN Wide Area Network. 25, 27
WiFi Wireless Fidelity. 8
WLAN Wireless Local Area Network. 20, 22, 25
WPA Wi-Fi Protected Access. 13
WWW World Wide Web. 3–5