Technische Grundlagen der Informatik

Transportschicht (Transport Layer)

Prof. Dr. phil. nat. habil. J. Haase, Dr.-Ing. Falko Schönteich

NORDAKADEMIE Hochschule der Wirtschaft

NORDAKADEMIE
HOCHSCHULE DER WIRTSCHAFT

Überblick

1 Einleitung

2 Segmente

3 Verbindungslose Übertragung

4 Verbindungsorientierte Übertragung

5 Network Address Translation (NAT)

Motivation

TeamViewer

ICQ

WhatsApp

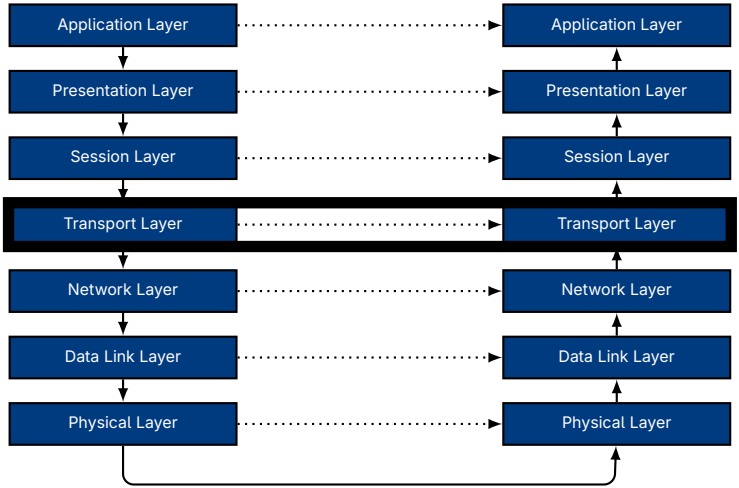
- Chat: Elektronische Kommunikation - meist über das Internet
- Benötigt man zum Übertragen der Nachrichten eine gesicherte oder ungesicherte Kommunikationsverbindung?

Bildquellen: https://de.wikipedia.org/wiki/TeamViewer, https://de.wikipedia.org/wiki/ICQ, https://de.wikipedia.org/wiki/WhatsApp

Lernziele

- Verständnis prinzipieller Eigenschaften von Diensten der Transportschicht
  - Multiplexing/Demultiplexing
  - Transportmechanismen zum zuverlässigen Transport von Daten
  - Flusskontrolle
  - Staukontrolle
- Kennenlernen von Internet-Protokollen der Transportschicht
  - User Datagram Protocol (UDP): verbindungslos
  - Transmission Control Protocol (TCP): verbindungsorientiert

Wiederholung



Was brauchen wir auf Transportschicht?

Aufgaben der Transportschicht

- (De-)Multiplexing von Datenströmen unterschiedlicher Anwendungen bzw. Anwendungsinstanzen
- zuverlässiger Transport von Daten (dies beinhaltet Sendewiederholung, Reihenfolgeerhaltung, Flusskontrolle, Staukontrolle)

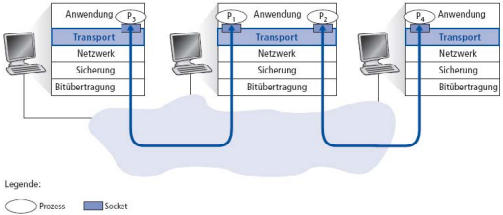
Multiplexing/Demultiplexing

Multiplexing beim Sender:

- Segmentierung von Datenströmen von Anwendungen wie Browser, Chat, E-Mail, …
- Segmente werden in jeweils unabhängigen IP-Paketen zum Empfänger geroutet

Demultiplexing beim Empfänger:

- Zuordnung der Segmente zu Datenströmen und Weiterleitung an Anwendung
- Adressierung der Anwendung?



Bildquelle: Folie 3-8, https://docplayer.org/15505387-Kapitel-3-transportschicht.html

Überblick

1 Einleitung

2 Segmente

3 Verbindungslose Übertragung

4 Verbindungsorientierte Übertragung

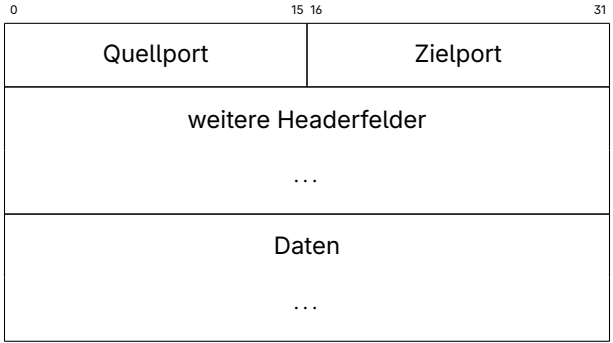
5 Network Address Translation (NAT)

Segment

Definition (Segment)

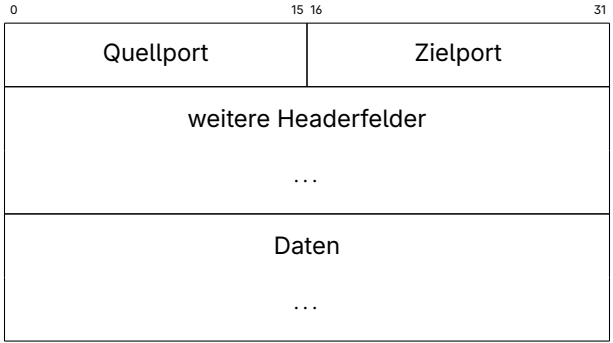
Ein Segment ist ein Paket der Transportschicht.

Hinweis: In der Literatur werden die Begriffe Segment und Datagramm teilweise unterschiedlich verwendet.



Segmentaufbau

1. Quellportnummer (Source-Port) - Welche Anwendung sendet die Nutzdaten?
2. Zielportnummer (Destination-Port) - Welche Anwendung soll die Nutzdaten verarbeiten?
3. Weitere Headerfelder: z. B. Prüfsumme, Transportprotokoll-Typ (TCP/UDP)
4. Anwendungsdaten



Ports?

Ports?

- Ports sind „Adressen" zur Adressierung einer Anwendung auf dem Host
- Ports sind 16-Bit Zahlen, die i. d. R. dezimal angegeben werden
- der Quellport eines Segments gibt den Port an, auf dem eine Anwendung eine Antwort erwartet (quasi die „Tür", durch die ein Paket ein Haus verlassen hat)
- Zielports sind quasi die „Eingangstür" der Zielanwendung

Well-known Ports

Viele der Ports 0 -1023 sind standardisierte Ports, welche auch als well-known Ports oder System Ports bekannt sind.

- Port 20: FTP - File Transfer Protocol, Datenverbindung (TCP)
- Port 21: FTP - File Transfer Protocol, Steuerverbindung (TCP)
- Port 22: SSH - Secure Shell
- Port 23: Telnet - Teletype Network (TCP)
- Port 25: SMTP - Simple Mail Transfer Protocol (TCP)
- Port 53: DNS - Domain Name System (TCP/UDP)
- Port 80: HTTP - Hypertext Transfer Protocol (TCP)
- Port 110: POPv3 - Post Office Protocol (TCP)
- Port 443: HTTPS - Hypertext Transfer Protocol over SSL/TLS[1] (TCP)

[1] Secure Sockets Layer/Transport Layer Security

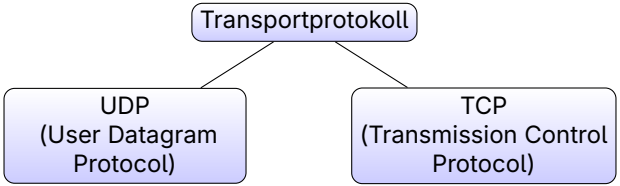
Well-known Ports

Registrierte Ports von 1024–49151 werden Benutzerprozessen bzw. Anwendungen zugeordnet.

- Port 2000: CISCO SCCP (Voice over IP) (UDP)
- Port 5004: RTP - Real-time Transport Protocol (UDP)
- Port 5050: Yahoo Messenger (TCP)
- Port 1863: MSN Messenger (TCP)
- Port 1433: MSSQL - Microsoft SQL-Server (TCP)
- Port 8008: POPv3 - Post Office Protocol (TCP)
- Port 1725: Valve Steam Client (UDP)

Dynamische oder private Ports 49152 - 65535 werden meist in dynamischer Form Clientanwendungen zugewiesen, sofern eine Verbindung hergestellt wird. Einige Anwendungen können sowohl TCP als auch UDP verwenden (z. B. DNS auf Port 53).

TCP/UDP



- verbindungslos (Datagramme werden unabhängig voneinander behandelt)
- verbindungsorientiert (reihenfolgeerhaltend, sichere Übertragung)

Überblick

1 Einleitung

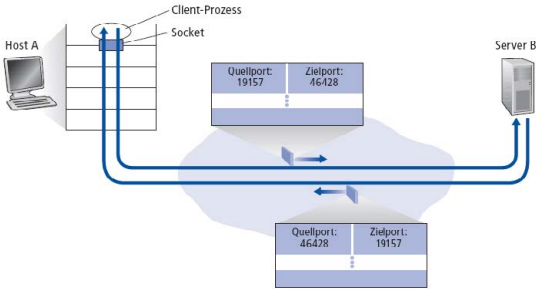
2 Segmente

3 Verbindungslose Übertragung

4 Verbindungsorientierte Übertragung

5 Network Address Translation (NAT)

Verbindungslose Kommunikation

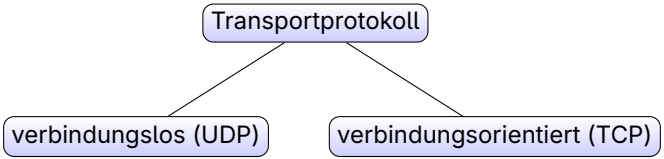


Bei der verbindungslosen Übertragung werden

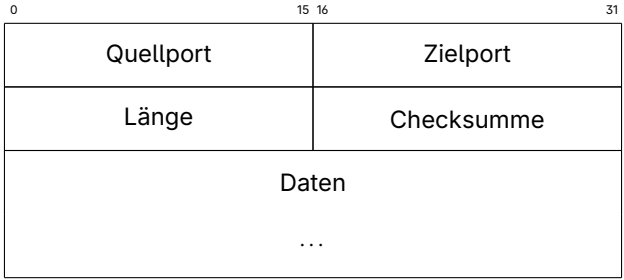
- Segmente unabhängig voneinander transportiert
- es gibt daher auch keine Garantie für eine korrekte Reassemblierung
- verlorene Segmente werden nicht wiederholt

Bildquelle: Folie 3-9, https://docplayer.org/15505387-Kapitel-3-transportschicht.html

TCP/UDP


Transportprotokoll
verbindungslos (UDP)
verbindungsorientiert (TCP)

User Datagram Protocol (UDP)



Der Aufbau des UDP-Headers ist sehr einfach, wodurch der Overhead für die Übertragung gering gehalten wird. UDP wird hauptsächlich für Anwendungen eingesetzt, bei denen Paketverlust in Kauf genommen werden kann, Latenz aber wichtig ist, z. B.:

- Voice over IP
- Videostreaming
- einige Online-Spiele

Diskussion: User Datagram Protocol (UDP)

Welche Vor- und Nachteile sehen Sie bei UDP ggü. einer verbindungsorientierten Kommunikation?

Vorteile:

Nachteile:

Überblick

1 Einleitung

2 Segmente

3 Verbindungslose Übertragung

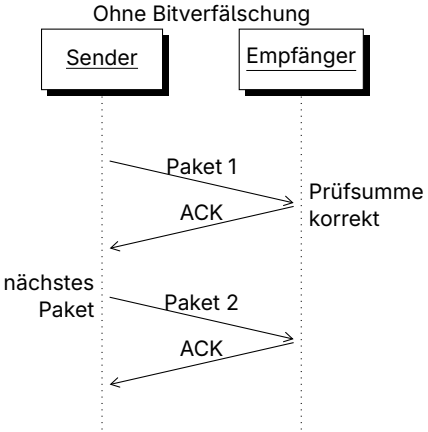
4 Verbindungsorientierte Übertragung

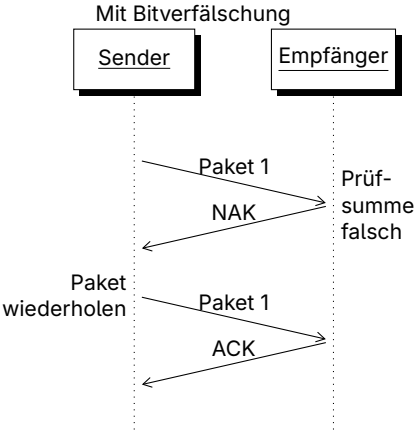
5 Network Address Translation (NAT)

Verbindungsorientierte Übertragung

Ziele

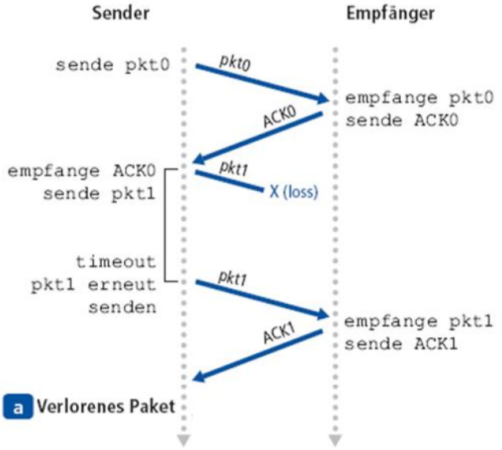
Sicherheit gegen Bitfehler?

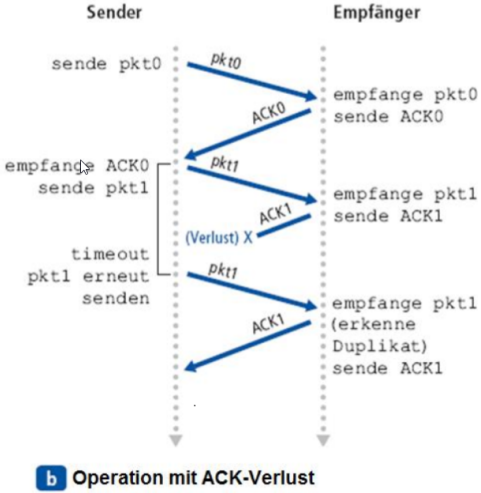




- ACK: Acknowledgement (Bestätigung)
- NAK: Negative Acknowledgement

Sicherheit gegen Paketverlust?





Bildquelle: Kurose/Ross, 241, modifiziert durch Kowalk

Problem: pkt1 kommt zweimal an

Wie können wir ohne NAK auskommen?

- Der Empfänger liefert im ACK die letzte Sequenznummer zurück, bis zu der alle Pakete erfolgreich empfangen wurden.
- Wird eine alte Sequenznummer empfangen, werden neuere Sequenznummern als nicht empfangen gewertet und neu gesendet

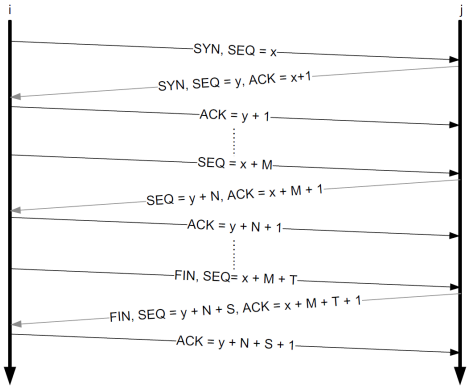
Verbindungsauf- und abbau

- Wie weiß der Empfänger, dass die erste empfangene Nachricht auch die erste gesendete ist?
- Wie weiß der Empfänger, dass er alles empfangen hat und seine letzte Bestätigung angekommen ist?
- **Anfang und Ende einer Verbindung muss von beiden Seiten bestätigt werden**

$\Rightarrow$ Drei-Wege-Handschlag

Drei-Wege-Handschlag

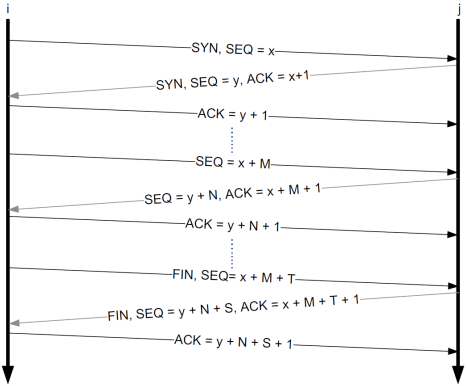
1. Verbindungsaufbau (Handshake): SYN
2. Datenübertragung: SEQ, ACK, (+Daten)
3. Verbindungsabbau (Teardown): FIN



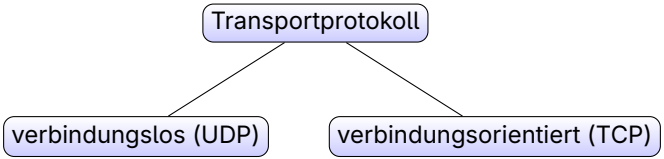
Drei-Wege-Handschlag

Performance der Datenübermittlung

- Die Datenübermittlung ist ineffizient, da stets nur ein Segment übertragen und quittiert werden kann
- Abhängig von der Rundlauf-Ausbreitungsverzögerung (*Round Trip Time* - RTT) wird die Bandbreite nicht nutzbringend ausgenutzt.



TCP/UDP


Transportprotokoll
verbindungslos (UDP)
verbindungsorientiert (TCP)

Transmission Control Protocol (TCP)

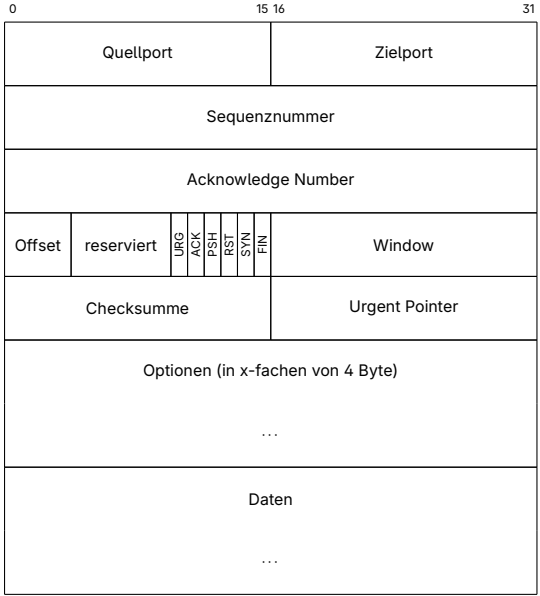
Das Transmission Control Protocol ist das dominierende Transportprotokoll im Internet. Es bietet:

- eine gesicherte, strom-orientierte (Byte per Byte) Übertragung von Daten
- Mechanismen zur Flusskontrolle d. h. der Empfänger steuert die Last
- Mechanismen zur Staukontrolle (*Congestion Control*) zur Vermeidung von Überlast im Netzwerk

Transmission Control Protocol (TCP)

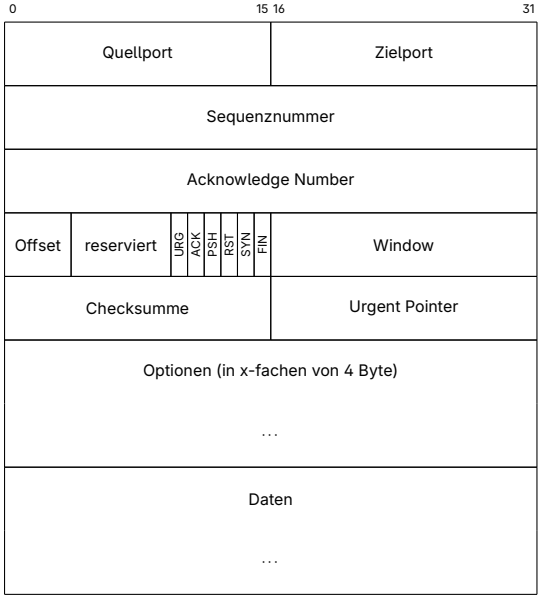
Header

- **Quellport** des Absenders
- **Zielport** des Empfängers, d. h. welcher Dienst des Empfängers wird adressiert.
- **Sequenznummer** zur gesicherten Übertragung
- **Acknowledge Number**: Quittierung empfangener Daten (welches Byte wird als Nächstes erwartet)



Transmission Control Protocol (TCP)

Header



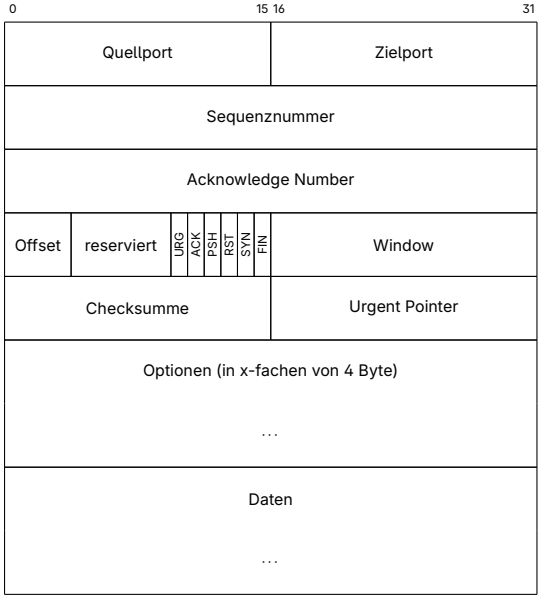
- **Offset** gibt die Länge des TCP-Headers in Vielfachen von 4 Byte an
- **reserviert** für spätere Erweiterungen
- **ACK**: Bestätigung einer empfangenen Nachricht

Anmerkung: Bestätigungen können *á la piggybacking* übertragen werden, d. h. die Bestätigung empfangener Daten von *A* nach *B* wird von *B* auch zum Senden eigener Daten nach *A* genutzt.

Transmission Control Protocol (TCP)

Header

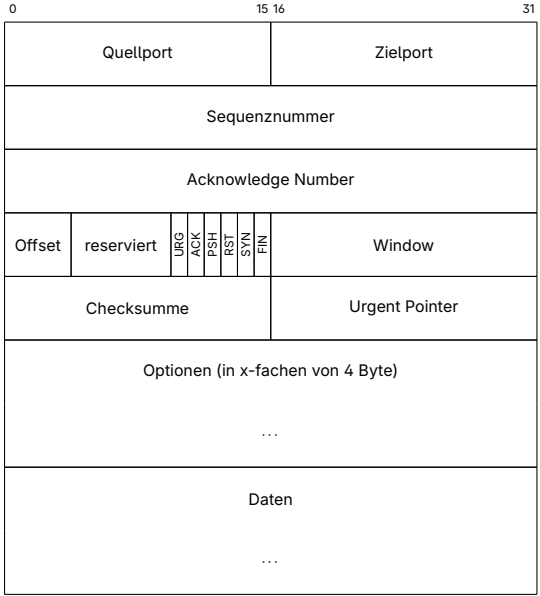
- **SYN**: Synchronize-Flag - Verbindungsaufbau
- **FIN**: Finish-Flag - Verbindungsabbau
- **PSH**: Push-Flag - falls gesetzt, werden sende- und empfangsseitige Puffer umgangen
- **RST**: Reset-Flag - Rücksetzen der Verbindung ohne Verbindungsabbau



Transmission Control Protocol (TCP)

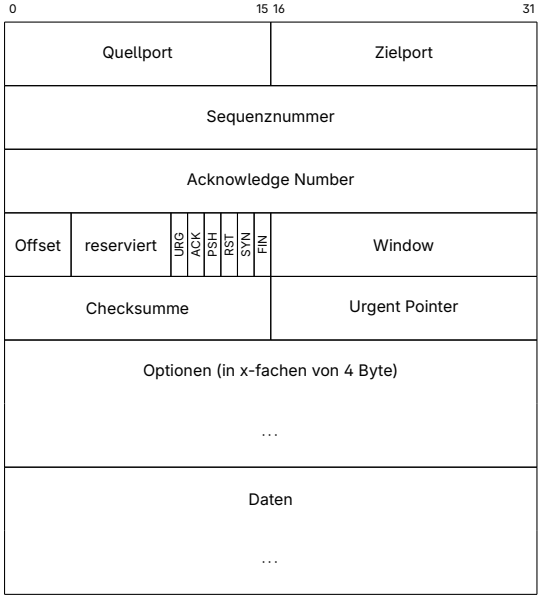
Header

- (Receive) **Window**: Größe des aktuellen Empfangsfensters W_r in Byte (Ermöglicht es dem Sender, die Datenrate zu reduzieren).
- **Checksumme** über Header und Daten
- **Urgent Pointer**: Gibt das Ende der Urgent-Daten an, welche unmittelbar nach dem Header beginnen und bei gesetztem URG-Flag schnell weitergereicht werden sollen.



Transmission Control Protocol (TCP)

Header



- zusätzliche **Optionen**, z. B.: Maximum Segment Size (MSS)

Beispiel:

- MSS bei Fast Ethernet: MTU (1500 Byte) - IP-Header (20 Byte) - TCP-Header (20 Byte) = MSS (1460 Byte)
- MSS bei DSL durch zusätzlichen PPPoE-Header: MSS = 1452 Byte

Praxisbeispiel Socket-Programmierung

Server

- Starten Sie Eclipse und erzeugen Sie ein neues Java-Projekt.
- Fügen Sie eine Serverklasse hinzu, die auf Verbindungen des lokalen Rechners auf Port 12345 warten soll und Daten, die dort ankommen auf der Konsole ausgibt; unter `https://gitlab2.nordakademie.de/snippets/5` finden Sie ein Beispiel für einen einfachen Server.
- Testen Sie diese Funktionalität mittels des Tools putty im Raw-Connection-Modus (oder telnet unter Linux).
- Testen Sie per netstat, ob Ihr Server wie erforderlich auf dem richtigen Port lauscht.
- Recherchieren Sie, was die folgenden Java-Befehle machen:
  - `new ServerSocket(12345);`
  - `server.accept();`
  - `client.getInputstream();`
  - `reader.readLine();`
- Sie haben ca. 30 Minuten, um Ihre Ergebnisse in Kleingruppen zu erarbeiten.

Praxisbeispiel Socket-Programmierung II

Client

- Fügen Sie eine Client-Klasse hinzu, die Daten sendet; unter `https://gitlab2.nordakademie.de/snippets/4` finden Sie ein Beispiel für einen einfachen Client.
- Testen Sie die Kommunikation
- Recherchieren Sie, wie ein GET-Request für einen Webserver aussieht und implementieren Sie einen einfach GET-Request in Ihrem Client. Testen Sie Ihren Client zuerst an folgendem Server im internen Netz der NORDAKADEMIE (VPN erforderlich oder Login in die VDI):
  `http://192.168.5.221:8080`
- Wenn alles funktioniert, können Sie ihn auch an einem anderen Webserver ausprobieren.
- Sie haben ca. 45 Minuten, um Ihre Ergebnisse in Kleingruppen zu erarbeiten.

Praxisbeispiel Socket-Programmierung III

Webserver

Recherchieren Sie, wie die TCP-Ausgabe eines Webservers nach einer Client-GET-Anfrage aussieht und implementieren Sie einen Webserver, der eine einfache HTML-Ausgabe zurückgibt. Testen Sie Ihren Server mit einem Webbrowser.

Überblick

1 Einleitung

2 Segmente

3 Verbindungslose Übertragung

4 Verbindungsorientierte Übertragung

5 Network Address Translation (NAT)

Network Address Translation (NAT)

Source NAT: Veränderung der Sendeadresse
Oftmals ist es notwendig, interne bzw. private IP-Adressen in einem Netzwerk auf eine oder mehrere global eindeutige Adressen abzubilden (z. B. auf die vom Provider zugewiesene öffentliche IP-Adresse).

Private IP:

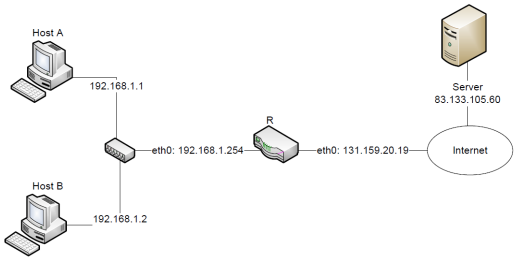
- Zur privaten Nutzung ohne vorherige Registrierung freigegeben
- Können mehrfach in jeweils unterschiedlichen Netzen vorkommen
- Sind somit nicht eindeutig und zur Adressierung im Internet i. d. R. nicht geeignet

Private Adressbereiche sind:

- 10.0.0.0/8
- 172.16.0.0/12
- 169.254.0.0/16
- 192.168.0.0/16

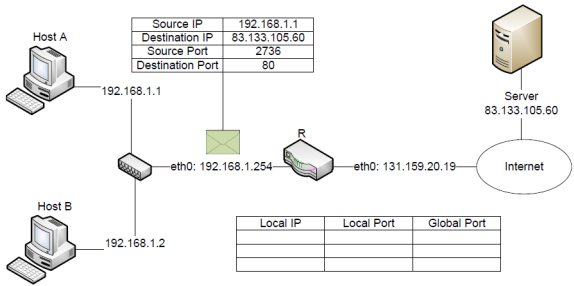
Beispiel 1.a

- Die Knoten Host A und Host B können im Netzwerk direkt über ihre privaten IP-Adressen kommunizieren.
- Der Router R ist über seine öffentliche Adresse 131.159.20.19 erreichbar. Host A und Host B sind von Hosts im Internet nicht erreichbar.



Beispiel 1.a

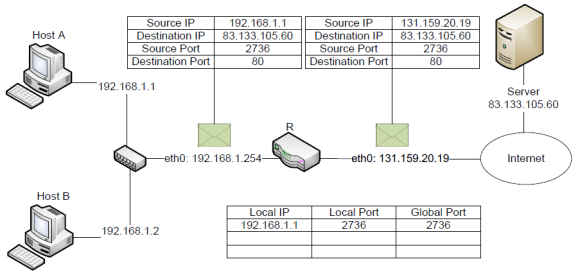
Host A will auf eine Webseite zugreifen, die auf dem Server mit der IP-Adresse 83.133.105.60 liegt und sendet ein Paket (TCP SYN) an den Server.



Der Quellport wird zufällig gewählt, der Zielport durch den *Application Layer* vorgegeben (Port 80 = HTTP).

Beispiel 1.a

Host A will auf eine Webseite zugreifen, die auf dem Server mit der IP-Adresse 83.133.105.60 liegt und sendet ein Paket (TCP SYN) an den Server.

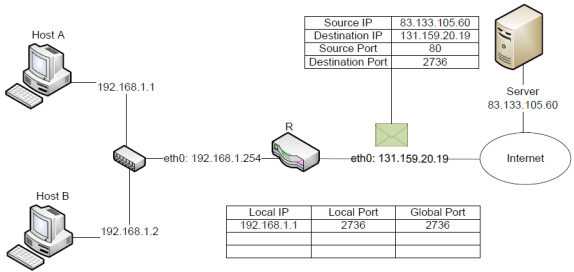


Die Adressübersetzung beim Router R erfolgt durch:

- R tauscht die Absenderadresse durch seine eigene, globale IP-Adresse aus
- R erzeugt einen Eintrag in seiner NAT-Tabelle, welche die Änderungen an dem Paket dokumentieren

Beispiel 1.a

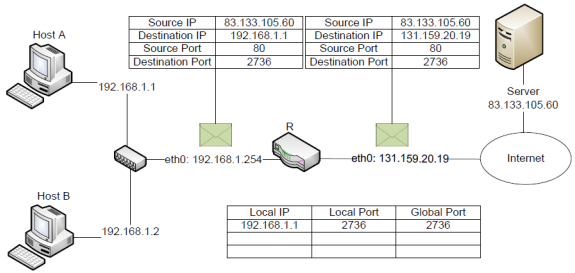
Der Server antwortet auf das empfangene Datenpaket und sendet die Antwort an den Router R, welcher für den Host A gehalten wird.



Im Allgemeinen „weiß" der Server nichts von der Adressübersetzung des Routers. Der Router R macht anschließend die Adressübersetzung wieder rückgängig und leitet das Paket an den Host A weiter. Auch der Host A „weiß" nichts von der Adressübersetzung.

Beispiel 1.a

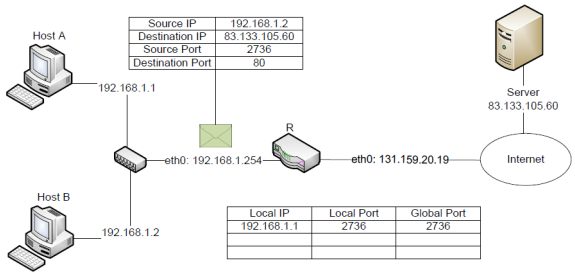
Der Server antwortet auf das empfangene Datenpaket und sendet die Antwort an den Router R, welcher für den Host A gehalten wird.



Im Allgemeinen „weiß" der Server nichts von der Adressübersetzung des Routers. Der Router R macht anschließend die Adressübersetzung wieder rückgängig und leitet das Paket an den Host A weiter. Auch der Host A „weiß" nichts von der Adressübersetzung.

Beispiel 1.b

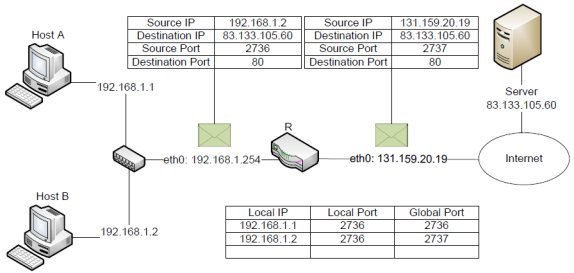
Nun will Host B ebenfalls auf den Server zugreifen und sendet ein SYN-TCP-Paket. Zufällig wählt Host B den gleichen Quellport wie Host A.



Der Router stellt fest, dass es bereits einen Eintrag in der NAT-Tabelle gibt, welcher eine IP-Adresse mit einem Quellport 2736 verknüpft. Deswegen generiert der Router einen neuen Eintrag mit einem inkrementierten globalen Quellport, so dass eine eindeutige Zuordnung möglich ist.

Beispiel 1.b

Nun will Host B ebenfalls auf den Server zugreifen und sendet ein SYN-TCP-Paket. Zufällig wählt Host B den gleichen Quellport wie Host A.



Aus Sicht des Servers wurden einfach zwei TCP-IP-Verbindungen vom Host R aufgebaut.