Infrastruktur Grundlagen

Transportlayerprotokolle / Flow Control

Transfer Control Protocol

- Teil der TCP/IP Protokollfamilie
- **Verbindungsorientiertes Protokoll**
- Garantiert eine **zuverlässige Verbindung** über ein **unzuverlässiges Netzwerk** (reliable VS unreliable)

TCP Multiplexing

- TCP verwendet sog. **Sockets**
- **Identifiziert End-to-End Verbindung**
- Auch **Virtual Ports** genannt
- **Physische NIC** kann **mehrere Verbindungen** herstellen

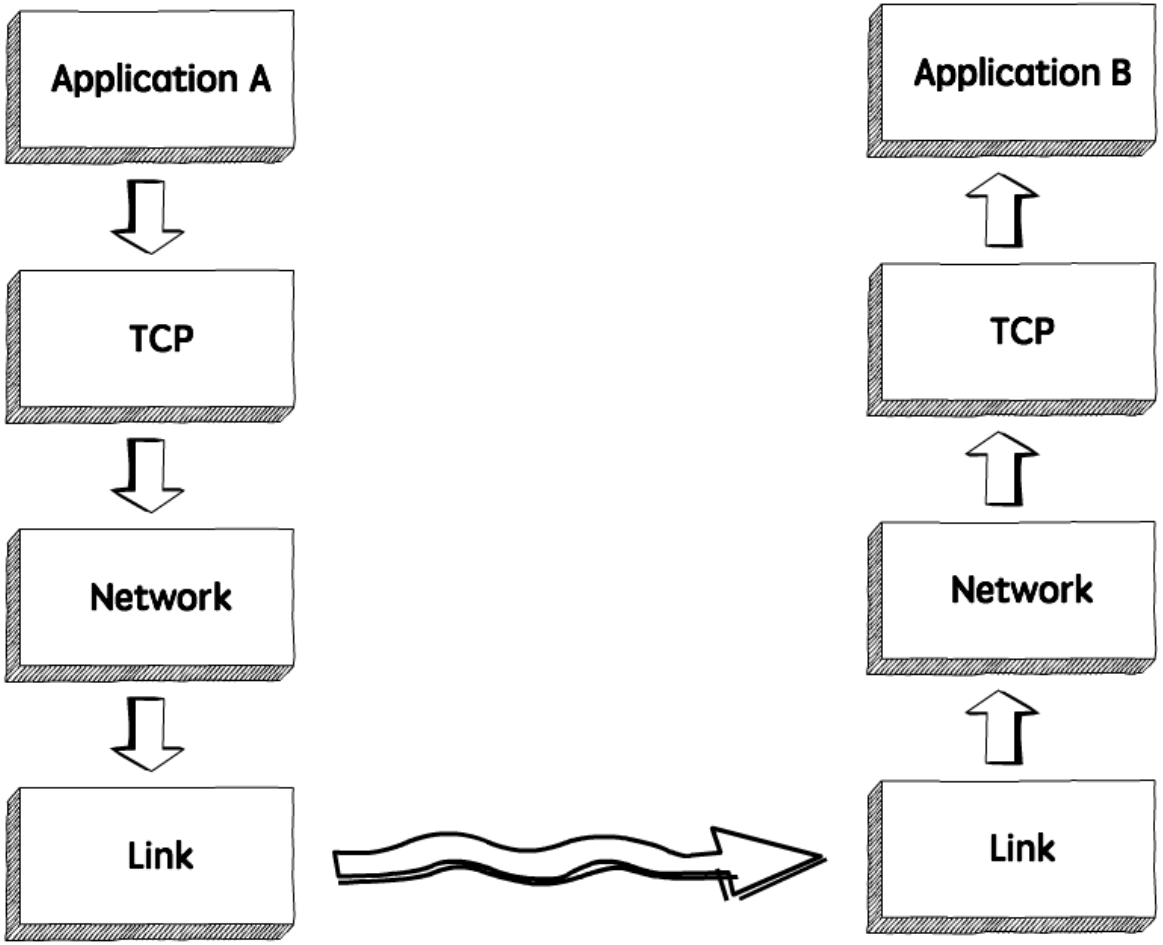
TCP Sockets

- **Socketverbindungen** werden über **Ports** identifiziert
- Bleiben während der Datenübertragung gleich
- IP + Port = Socket Endpoint
- Beispielsweise:
  - www.orf.at:80
  - 127.0.0.1:22

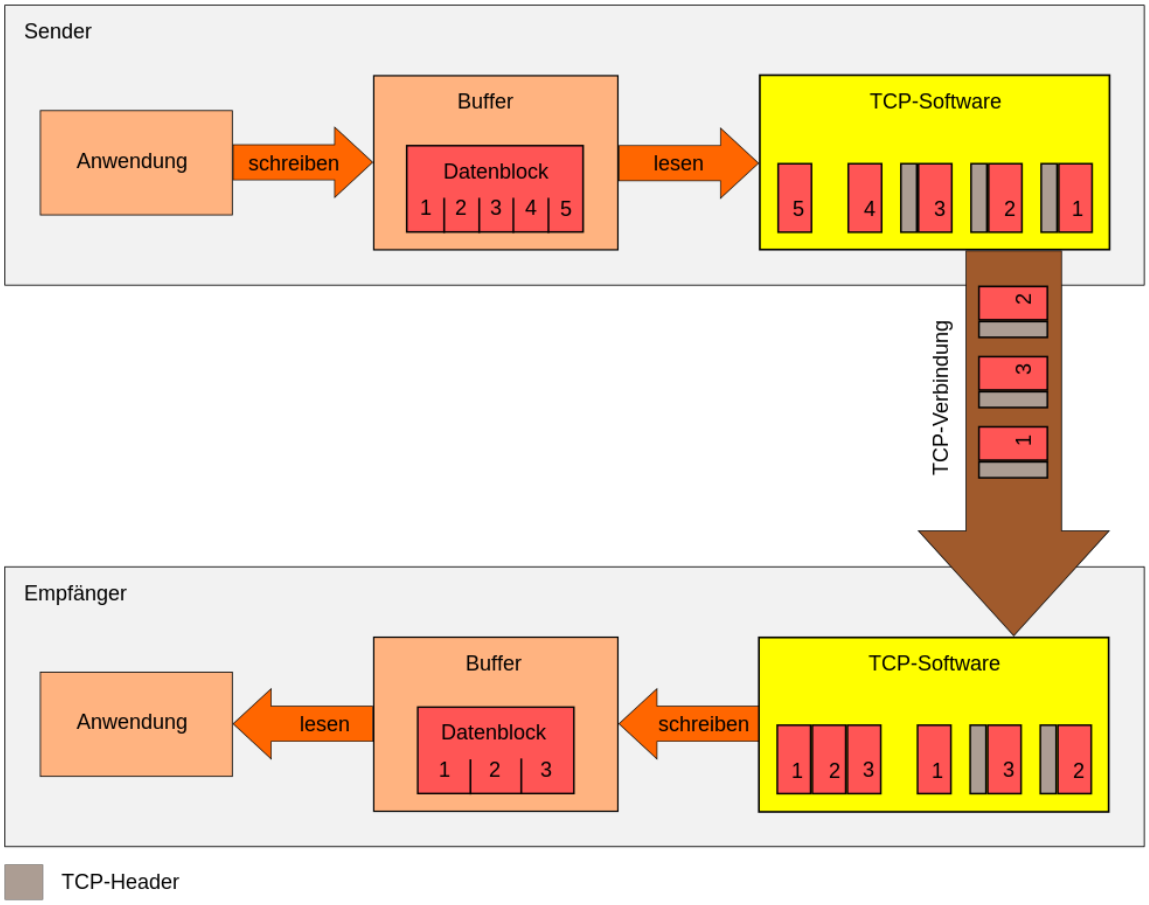
TCP Datenübertragung

- Nimmt Daten vom **Application Layer** entgegen
- TCP-**Sender teilt Daten** in **Segmente**
- TCP-**Empfänger assembliert Segmente** zu **Data Streams**
- Data Stream wird an den **Application Layer weitergereicht**

TCP Verbindung



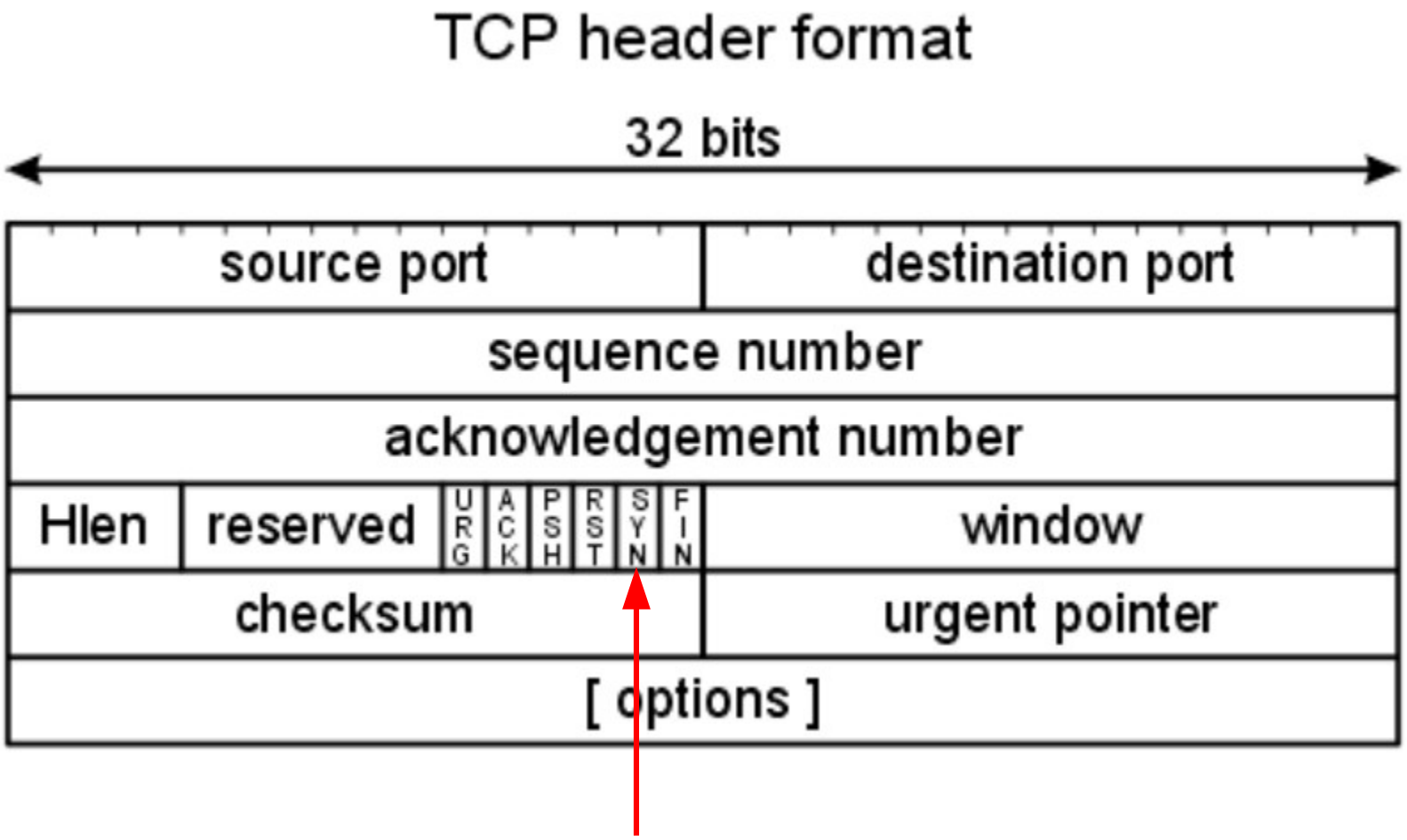
TCP Übertragung



TCP Assemblierung

- **Empfänger assembliert** empfangene Segmente
- Assemblierte Segmente ➔ **Data Stream**
- Data Stream wird an Applikation übergeben

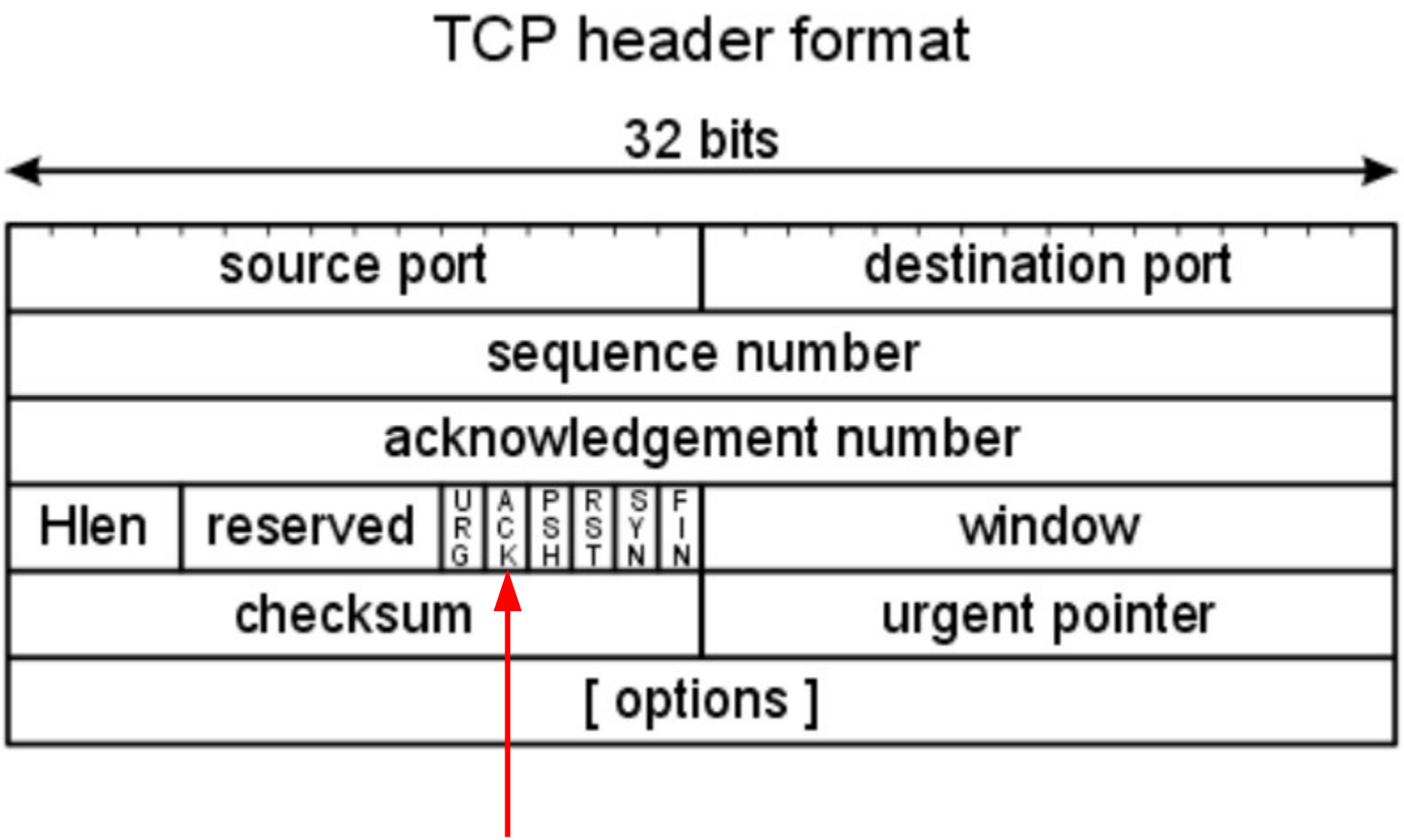
SYN im TCP-Header



SYN-Flag im TCP-Header

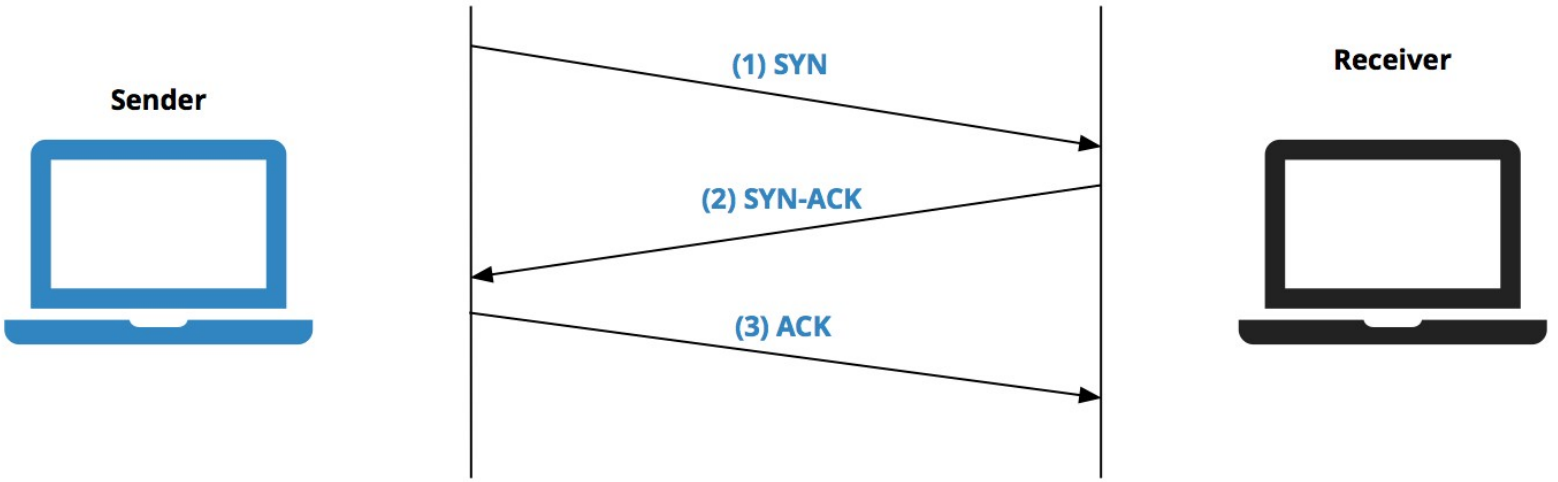
- **Synchronize-Flag**
- **Initiiert** eine TCP-Verbindung
- Server antwortet im Normalfall mit **SYN-ACK** als **Bestätigung**
  - Anderenfalls mit RST
- **Synchronisierung** von **Sequence Numbers**
  - notwendig für die Fehlererkennung
  - Wireshark zeigt relative Zahl (beginnt bei 0)
  - 32 bit Wert
    - – 4,294,967,296 Möglichkeiten
  - **I**nitial **S**equence **N**umber (**ISN**)
    - Wird vom Betriebssystem generiert

ACK im TCP-Header

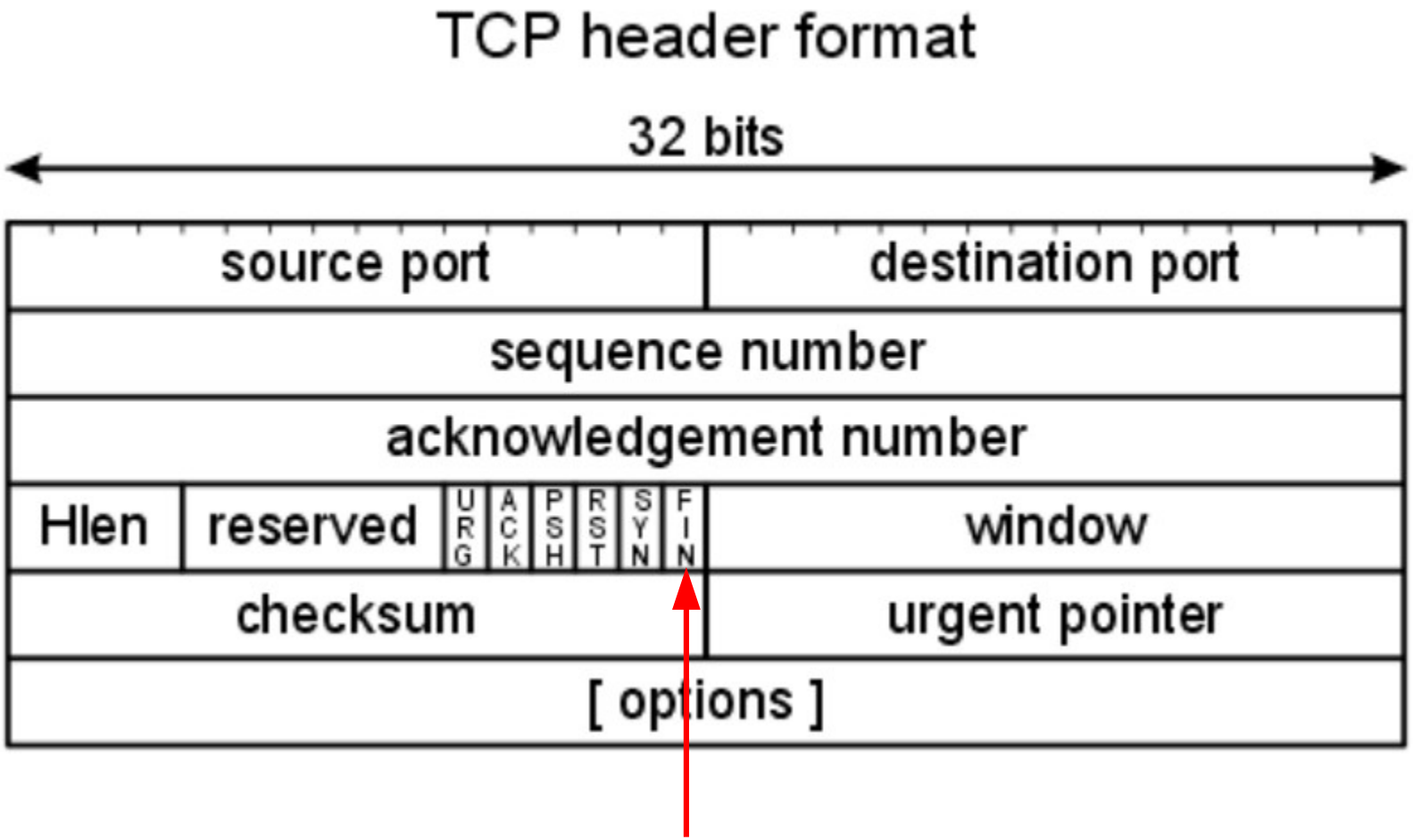


ACK-Flag im TCP-Header

- Acknowledgment-Flag
- **Bestätigt** den **Empfang einer Nachricht**
- Beispiel: TCP 3-Way-Handshake:



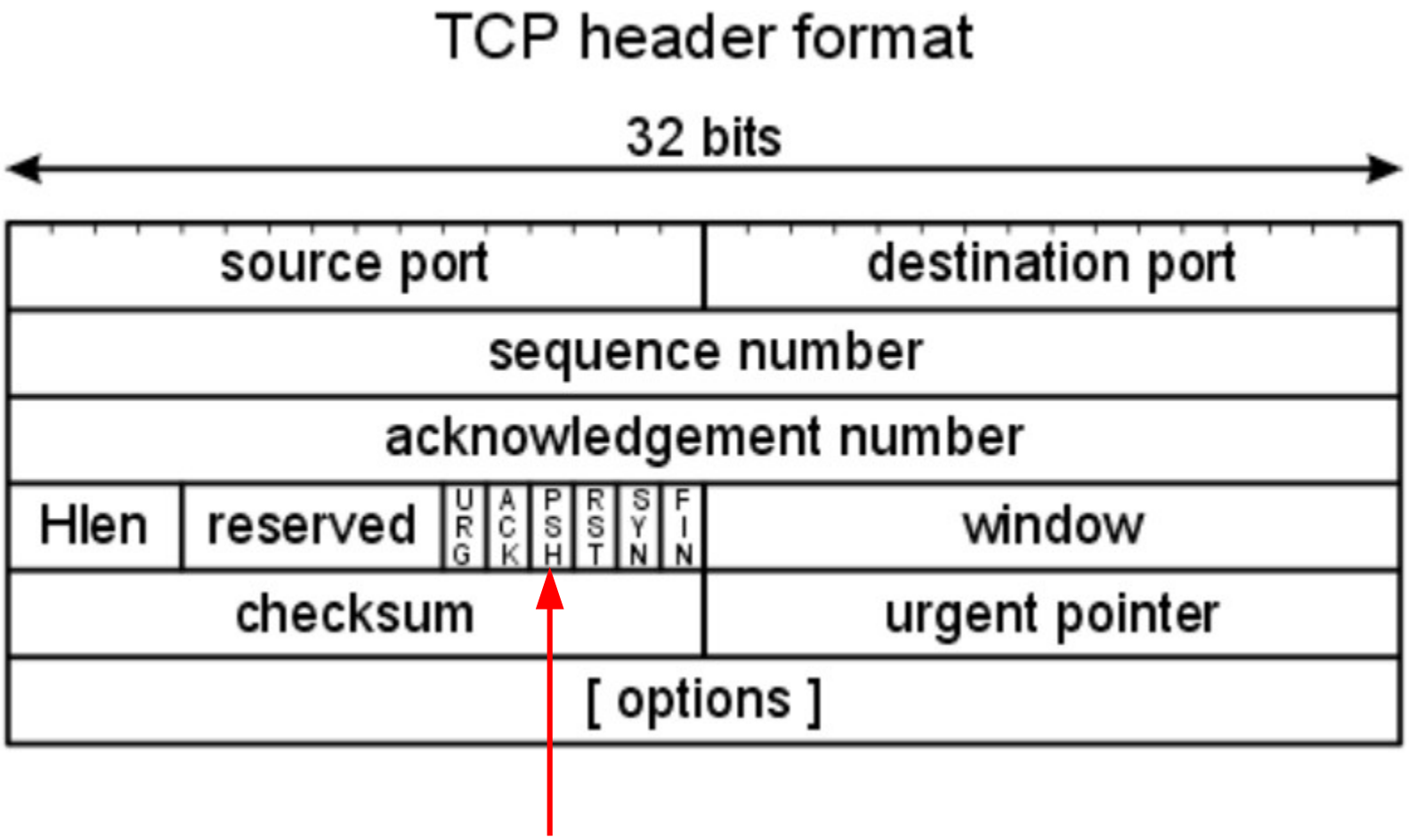
FIN im TCP-Header



FIN-Flag im TCP-Header

- **Schlussflag** (finish)
- Signalisiert dem TCP-Gegenüber einen **Verbindungsabbau**
- Dient zur Freigabe der TCP-Verbindung
- Zeigt dem Empfänger, dass keine weiteren Daten vom Sender kommen

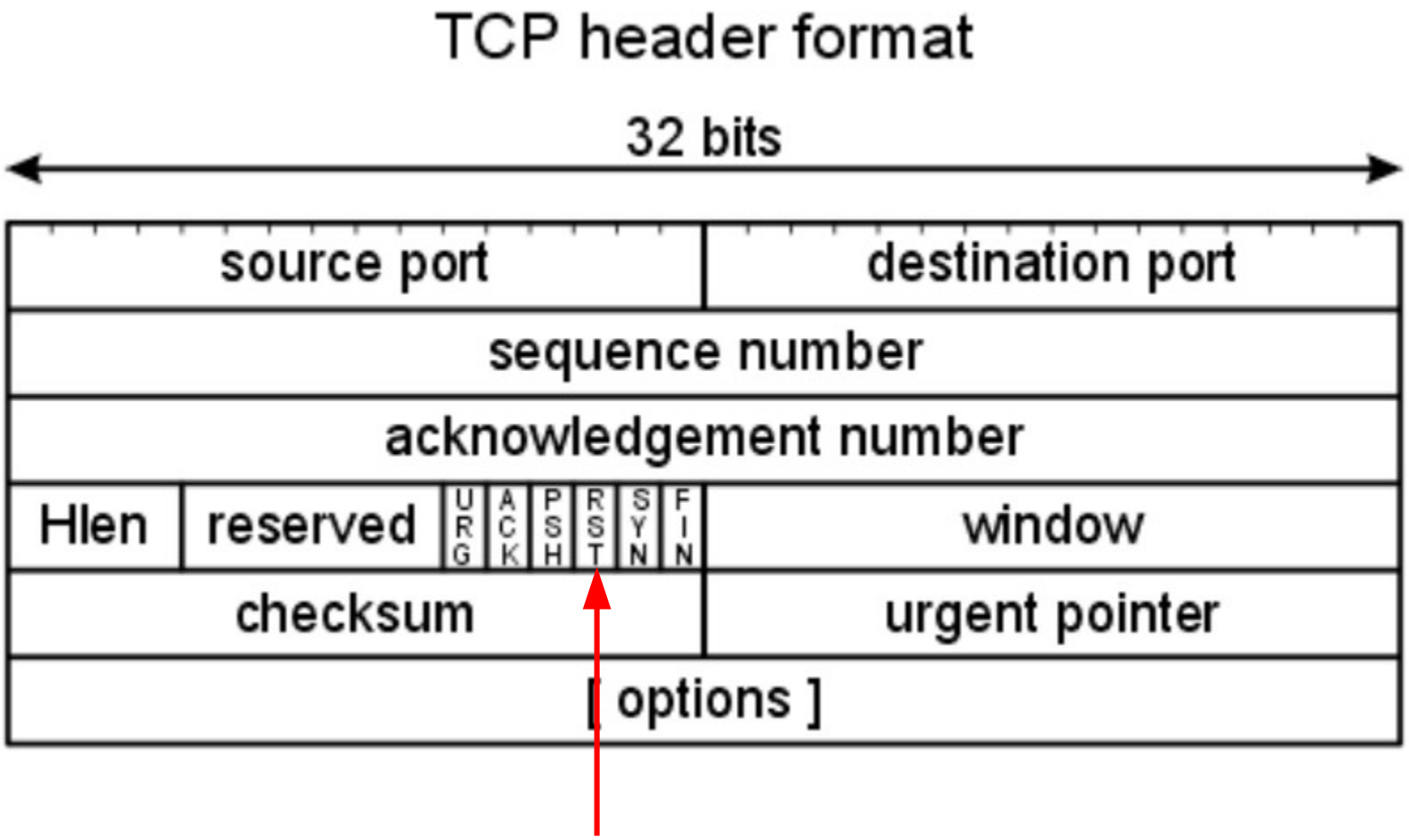
PSH-Flag im TCP-Header



PSH-Flag im TCP-Header

- **PUSH-Flag** (PSH Bit)
- **Empfänger** sammelt Daten im **TCP Buffer**
- Ist der Buffer voll, werden die Daten an die Applikation übertragen
- **Problem**: **Echtzeichtapplikationen** (z.b.: telnet, ssh)
- **PSH-Flag** sorgt dafür, dass die **Daten sofort** an die **Applikation weitergegeben** werden (unabhängig vom Füllstand des Buffers)
- **Umgeht** damit den **Buffer** von **Sender** und **Empfänger**

RST im TCP-Header

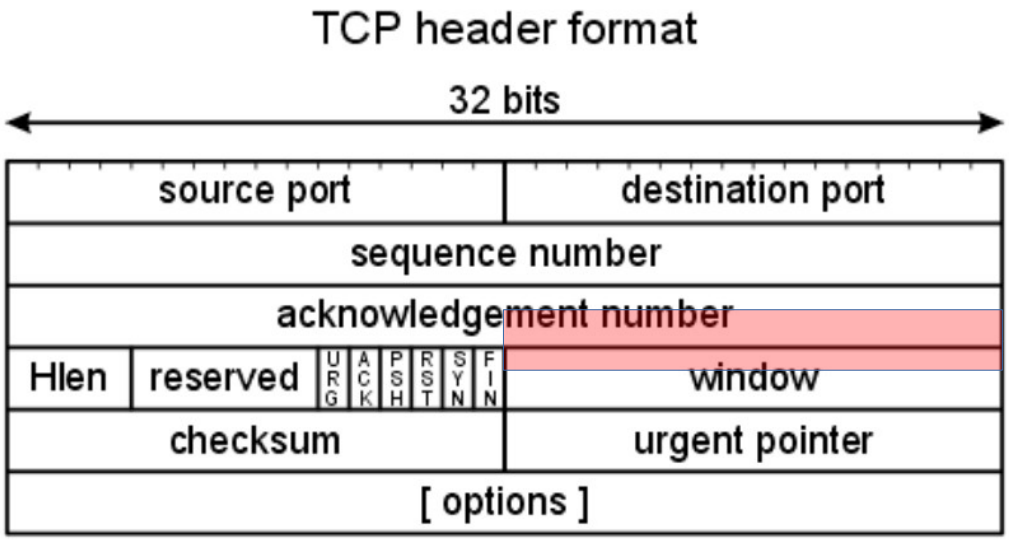


RST-Flag im TCP-Header

- **Reset Flag** (RST Bit)
- Wird gesetzt **wenn** eine **Verbindung abgebrochen** werden **soll**
  - Port **nicht erreichbar**
  - **Technische Probleme**
  - **Abweisung unerwünschter Verbindungen**

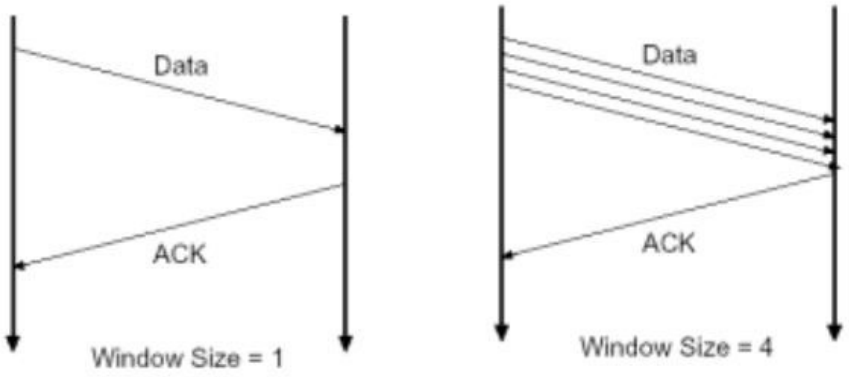
TCP Window Size

- **„Receive Window“** - Buffer des Empfängers
- Window Size bestimmt die Menge an Paketen bis ACK gesendet wird
- **Maximale Datenmenge** die ein Host **auf einmal empfangen kann**



Window Size & ACK

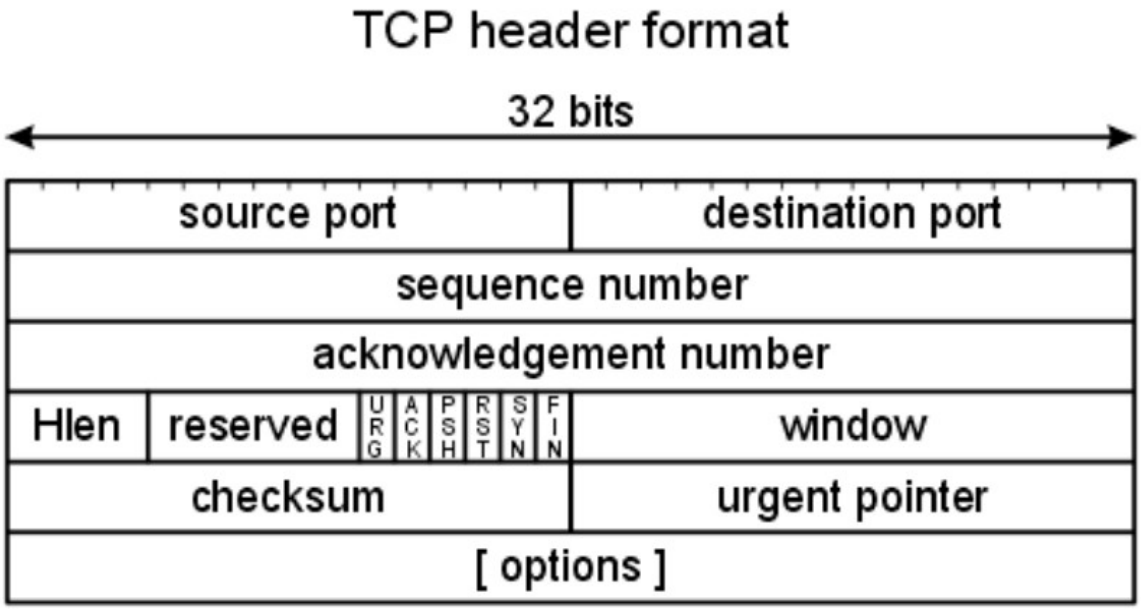
- Stellt sicher, dass der **Empfängerbuffer** nicht überfordert wird
  - **Kleines Fenster** ➔ Sender muss oft auf ACK warten
  - **Großes Fenster** ➔ Im Fehlerfall müssen mehr Daten erneut übertragen werden



- Empfänger sendet **Window Size 0** wenn der sein **Buffer voll** ist
- Sender versucht nach Wartezeit erneut zu senden

TCP Sequence / Acknowledgement

- **Sequence & Acknowledgment-Numbers**
  - **Nummerieren** das Senden und Empfangen von Daten
  - Stellen sicher, dass alle Pakete übertragen wurden



TCP Sequence Number

- **I**nitial **S**equence **N**umber (**ISN**)
  - Wird vom Betriebssystem generiert
- 32 bit Wert
  - 4,294,967,296 Möglichkeiten
- Wireshark zeigt relative Zahl (beginnt bei 0)

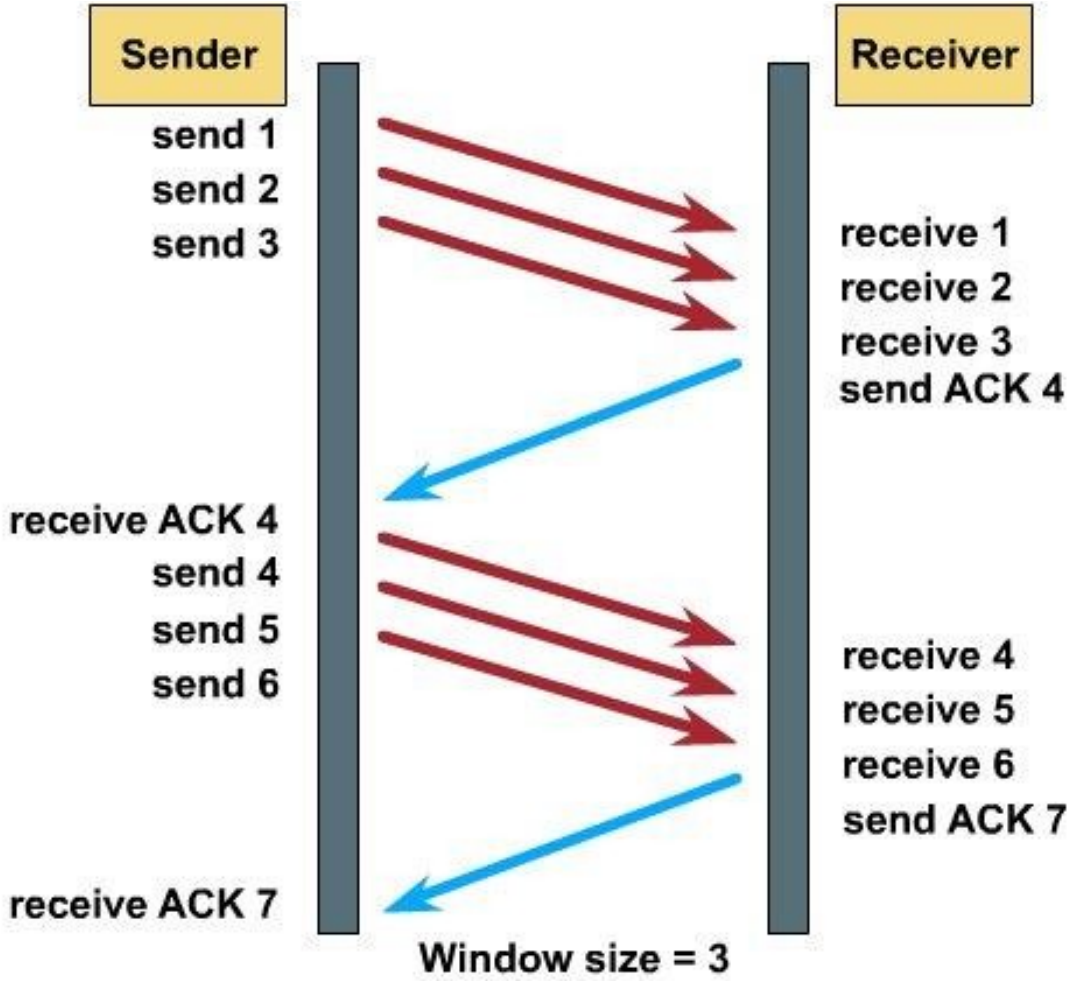
TCP Acknowledgement Number

- **Empfänger sendet ACK Number an** den **Sender**
- **Bestätigt fehlerfrei empfangene Datenpakete**
- Empfangene Bytes + 1
  - 206 Bytes empfangen ➔ ACK = 207
- Gibt an welche Sequence Number als Nächste vom Sender erwartet wird ➔ Seq=207

TCP Flow Control

- **Zuverlässige Datenzustellung in unzuverlässigen Netzen**
- **Sichert** die **Einhaltung** der Übertragungs**reihenfolge**
- **Sender sendet nicht mehr Daten als Empfänger verarbeiten kann**
  - **➔ Window Size im TCP-Header**
- Sender kann mehrere Datenpakete ohne Verzögerung senden
- **Sliding Window Protokoll**
  - Flusssteuerung - Verhindert Datenstau
  - Zuverlässige Übertragung von Paketen

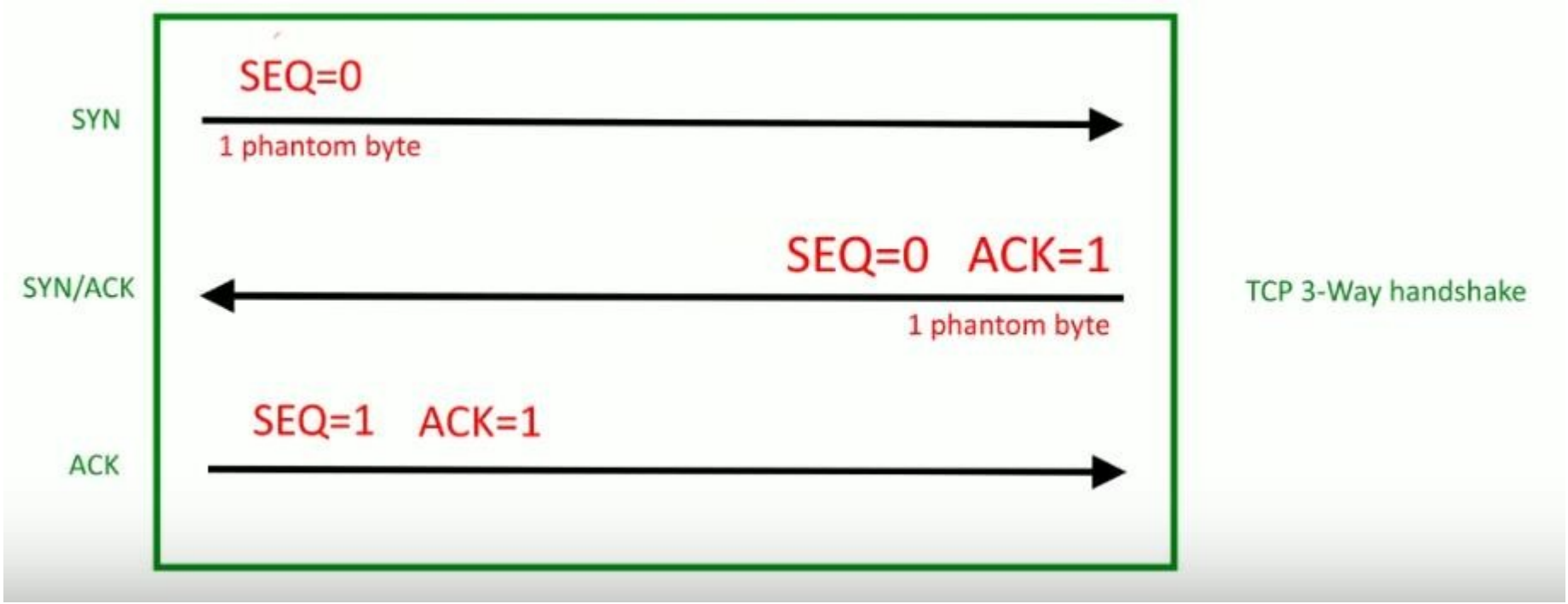
TCP Sliding Window Beispiel



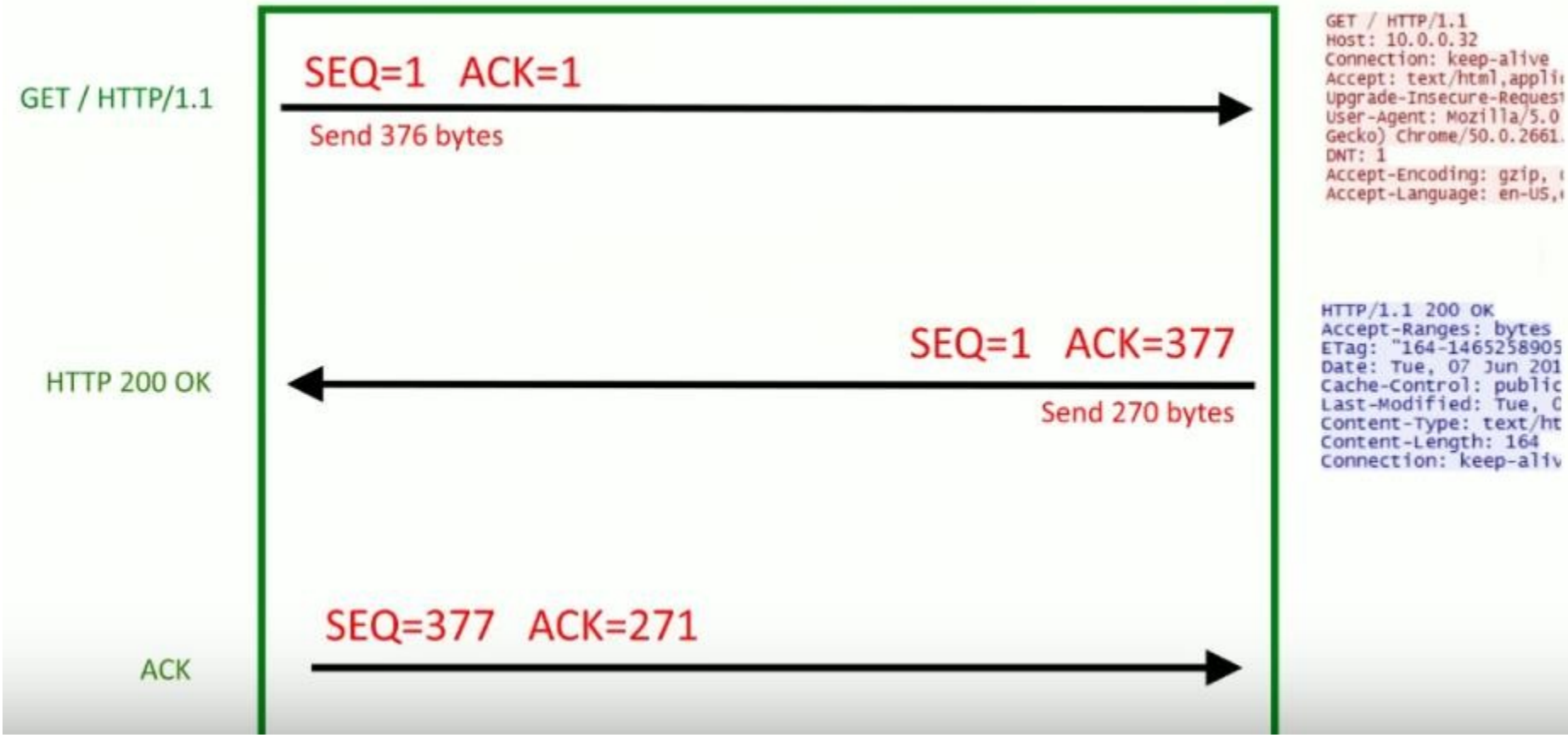
TCP Sliding Window

- **Empfänger bestimmt Anzahl der Pakete** (10 – Window Size)
- Sender übermittelt Paket 1 bis 10
- Paket 5 geht auf dem Transportweg verloren
- Empfänger meldet, dass Paket 5 nicht angekommen ist
  - ACK = 5
- Sender passt Sliding Window an und sendet Paket 5 bis 10 erneut
- Empfänger übernimmt Pakete 5 bis 10 erneut

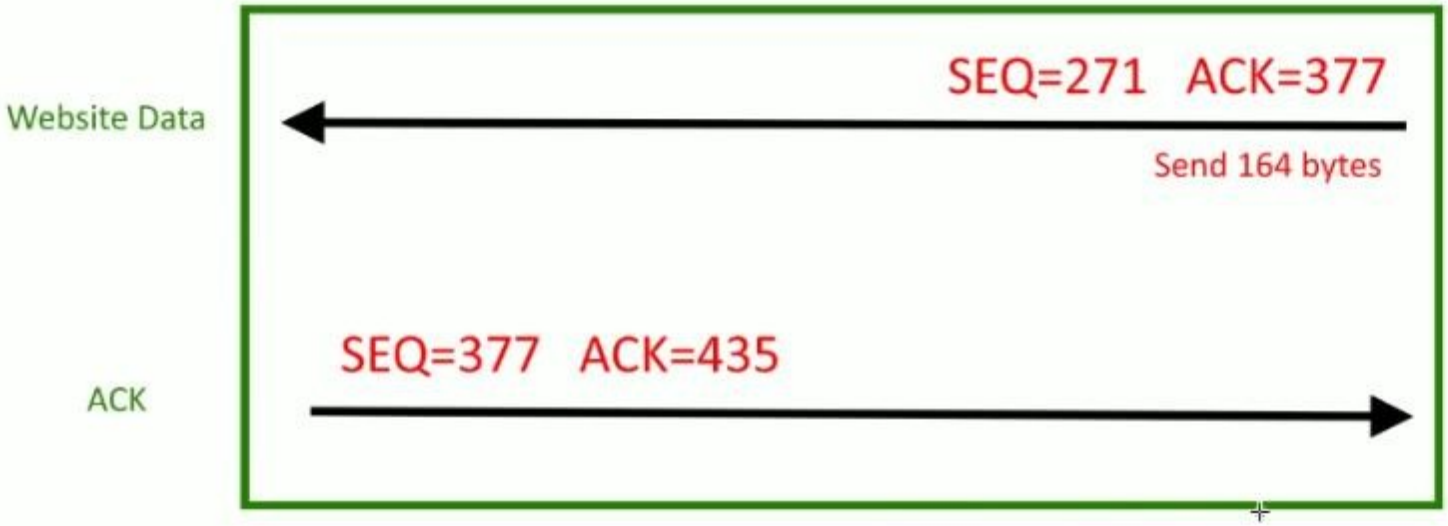
TCP Verbindungsaufbau



TCP Datenübertragung



TCP Datenübertragung

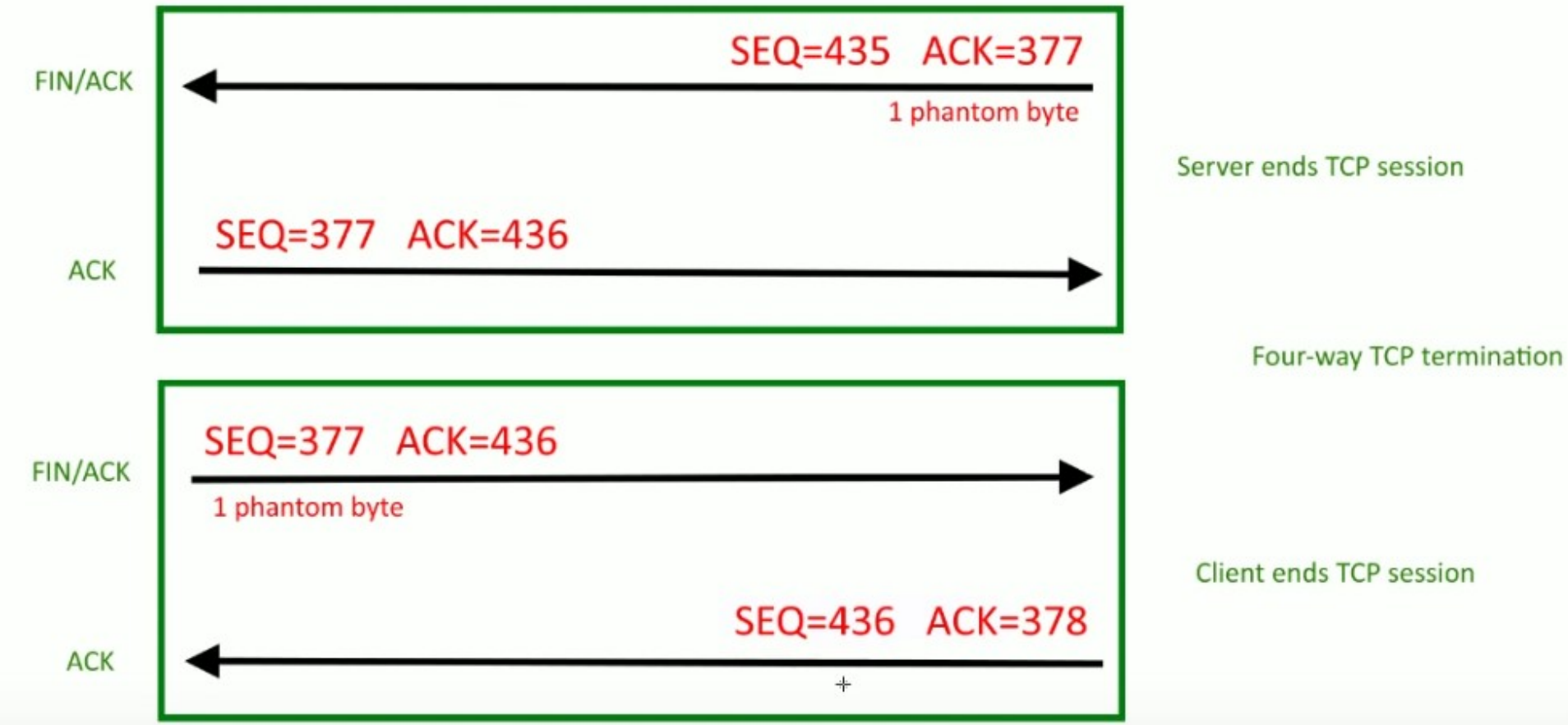



```
<!DOCTYPE html>
<html lang="en">
  <head>

    <title>Derpy Net
 <body>

 <H1> Derpy </H1>
<!--
```

TCP Verbindung beenden



TCP Retransmission Timer

- Sender verwendet Timeout bis ACK vom Empfänger kommt
- Erhält der Sender kein ACK in definierter Zeit so sendet er die Pakete erneut
- **Zu niedriger Timeout** ➔ Korrekte Pakete werden erneut gesendet
- **Zu hoher Timeout** ➔ Verlorenes Paket wird unnötig spät gesendet
- Berechnet sich über die **RRT** (**R**ound **T**rip **T**ime)
  - ➔ Gemessene Durchschnittszeit für die Übertragung von Paketen

UDP - Eigenschaften

- **U**ser **D**atagram **P**rotocol
- **Verbindungsloses Transportprotokoll**
- Arbeitet wie TCP auf dem **OSI Layer 4** (=Transportschicht)
- besitzt keine Datenflusskontrolle

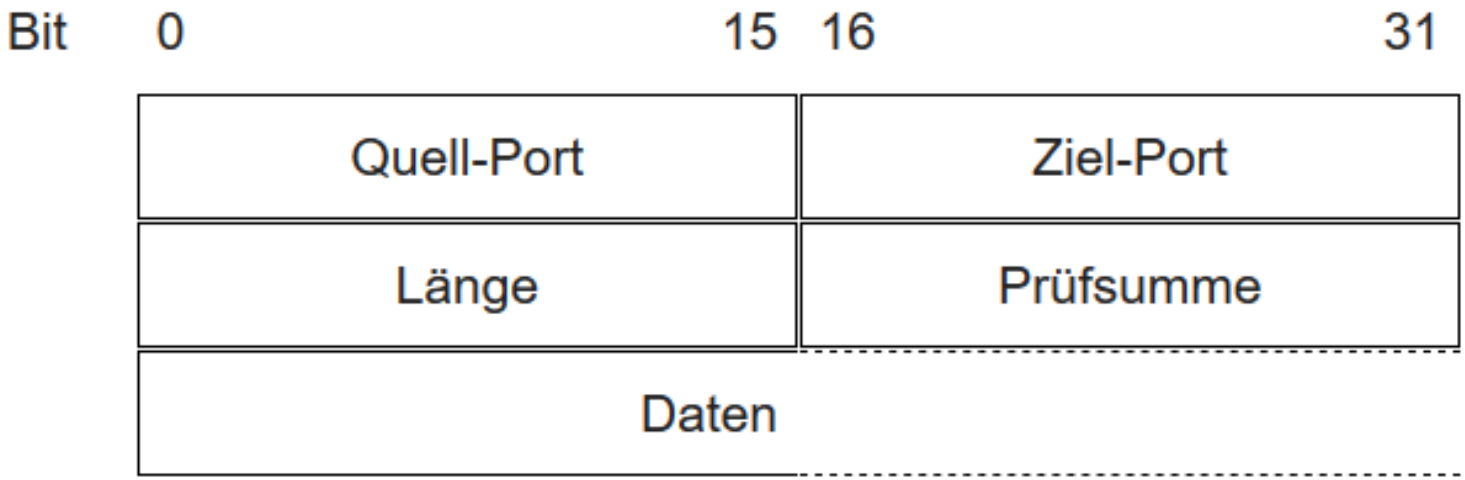
- **Vorteile**:
  - **UDP-Header** ist mangels Datenflusskontrolle **kleiner**
  - **Einfacher** zu implementieren
  - für „**Echtzeit**“ **Anwendungen** geeignet (z.b. Video/Audio Streams)
- **Nachteile**:
  - Kein Verbindungsmanagement
  - Keine Flusskontrolle
  - Keine Zeitüberwachung
  - Keine Fehlerbehandlung

UDP - Funktionsweise

- Ähnlich TCP **ohne Kontrollfunktionen**
- verwendet wie TCP eine **Portstruktur**
- UDP ist **nicht in der Lage** den **Datenstrom zusammen zu setzen**
  - Empfangende **Anwendung kümmert** sich um das **Datenmanagement**:
    - **Paketreihenfolge**
    - **Fehlererkennung**
- **Verbindungsstatus nicht verfügbar** (wegen fehlendem 3-Way Handshake)

UDP Header

- Headerlänge 64 Bits ➔ 8 Byte



UDP-Datagramm Header Format

UDP Ports Beispiele

Portnummer	Protokoll	Anwendung
53	DNS	Domain Name System
69	TFTP	Trivial File Transfer Protocol
67 & 68	DHCP	Dynamic Host Configuration Protocol
161	SMNP	Simple Managment Network Protocol

UDP Applikationen

- **Audio / Video Streaming**
  - Videokonferenz Tools
  - **V**oice **o**ver **IP** (VOIP)
  - TV **Streaming** (z.b. über Kabelnetzbetreiber)
- **Computerspiele**
  - Anforderung auf **Echtzeit**
- **DNS Server Lookups** (kann auch als TCP durchgeführt werden)
  - Domain Name Service

Vergleich TCP / UDP

- **TCP**
  - zuverlässig
  - geordnet
  - „schweres“ Protokoll
- **UDP**
  - unzuverlässig
  - ungeordnet
  - „Leichtes“ Protokoll