



# Transport Layer

---

## ISO/OSI Referenzmodell Schicht 4

## TCP/IP Referenzmodell Schicht 3

OSI-Referenzmodell	TCP/IP-Schichtenmodell
Anwendungsschicht (Application layer)	Anwendungsschicht
Darstellungsschicht (Presentation layer)	
Sitzungsschicht (Session layer)	
Transportschicht (Transport layer)	Transportschicht
Vermittlungsschicht (Network layer)	Netzwerkschicht
Sicherungsschicht (Data link layer)	Verbindungsschicht
Bitübertragungsschicht (Physical layer)	



# Transport Schicht

---

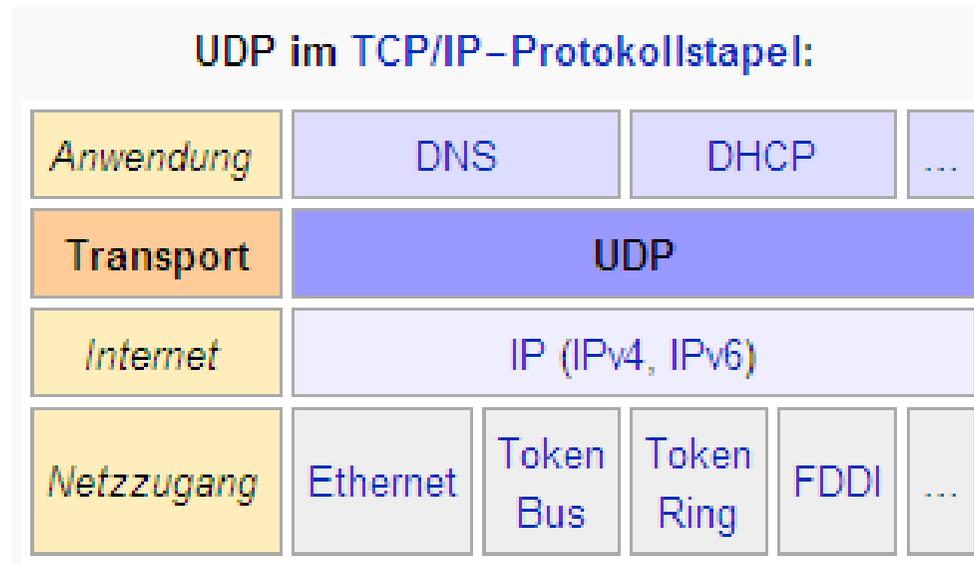
- o **User Datagram Protocol (UDP)**
- o **Transmission Control Protocol (TCP)**



# UDP User Datagram Protocol

---

- o Verbindungslose Übertragung von Daten über das Internet
- o RFC 768 (1980)



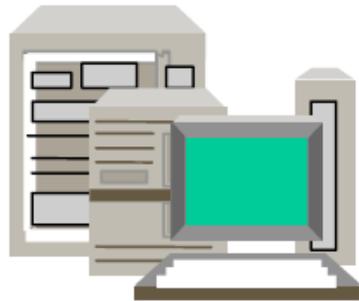
Quelle: [http://de.wikipedia.org/wiki/User\\_Datagram\\_Protocol](http://de.wikipedia.org/wiki/User_Datagram_Protocol)



# Portnummern

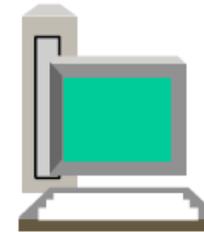
---

## Portnummer (kurz Ports)

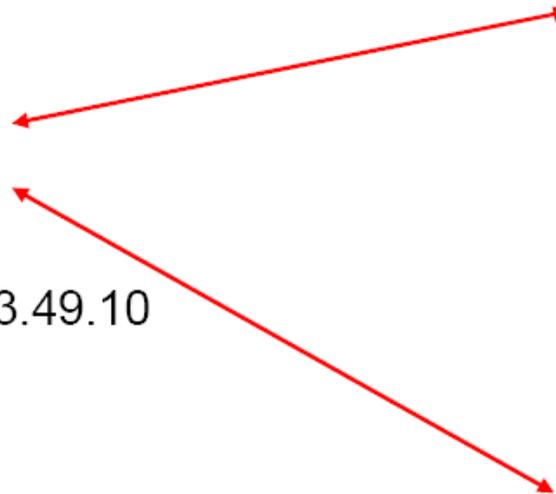
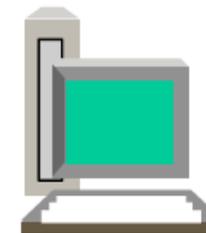


www-Server, 143.93.49.10  
Standard port: 80

www-Client, 143.93.49.23  
port: 1027



www-Client, 143.93.49.53  
port: 1029

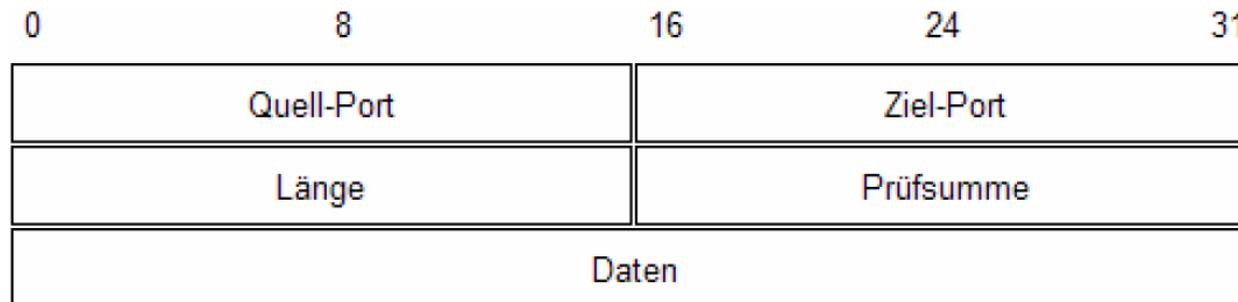




# UDP User Datagram Protocol

---

## UDP Header



Quelle: [http://de.wikipedia.org/wiki/User\\_Datagram\\_Protocol](http://de.wikipedia.org/wiki/User_Datagram_Protocol)

**Quell-Port (Source Port)** Enthält die optionale Adresse des Sende-Ports. Bei Antworten auf Datenpakete kann durch die Portadresse der jeweilige Prozess unmittelbar wieder angesprochen werden. Wird vom Sender kein Sende-Port definiert, so wird diese Feld mit dem Wert „0“ übertragen.

**Ziel-Port (Destination Port)** Enthält die Adresse des Empfänger-Ports

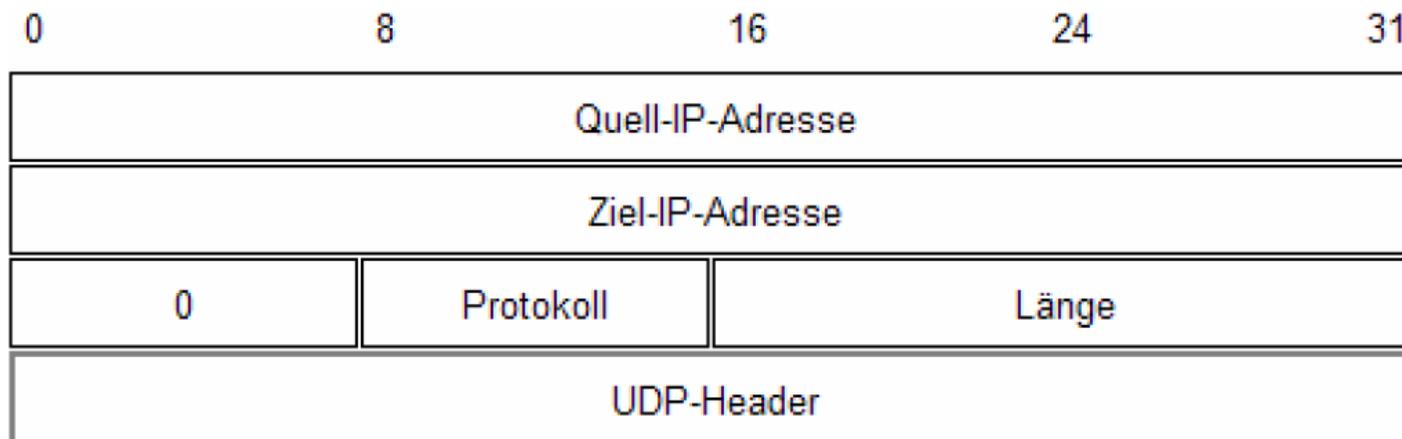
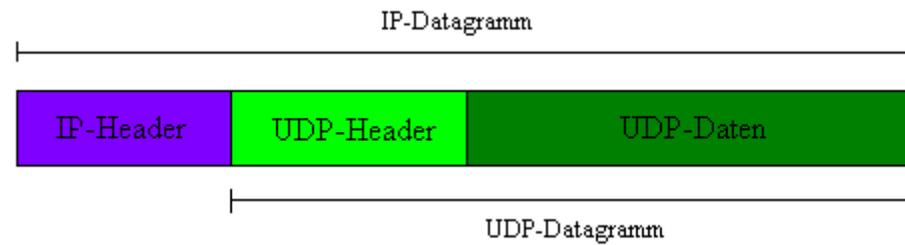
**Länge (Length)** Definiert die Gesamtlänge des Datenpaketes, inklusive Header und Nutzdaten

**Prüfsumme (Checksum)** Enthält eine optionale Prüfsumme. Der Wert „0“ weist darauf hin, dass keine Berechnung erfolgt ist. Die Prüfsumme wird aus dem UDP-Header und einem 96-Bit-Pseudo-Header errechnet.



# UDP und IP Header / Pseudo-Header

Teile des IP-Headers werden zur Erzeugung der UDP-Prüfsumme in den sogenannten „Pseudo-Header“ übernommen.

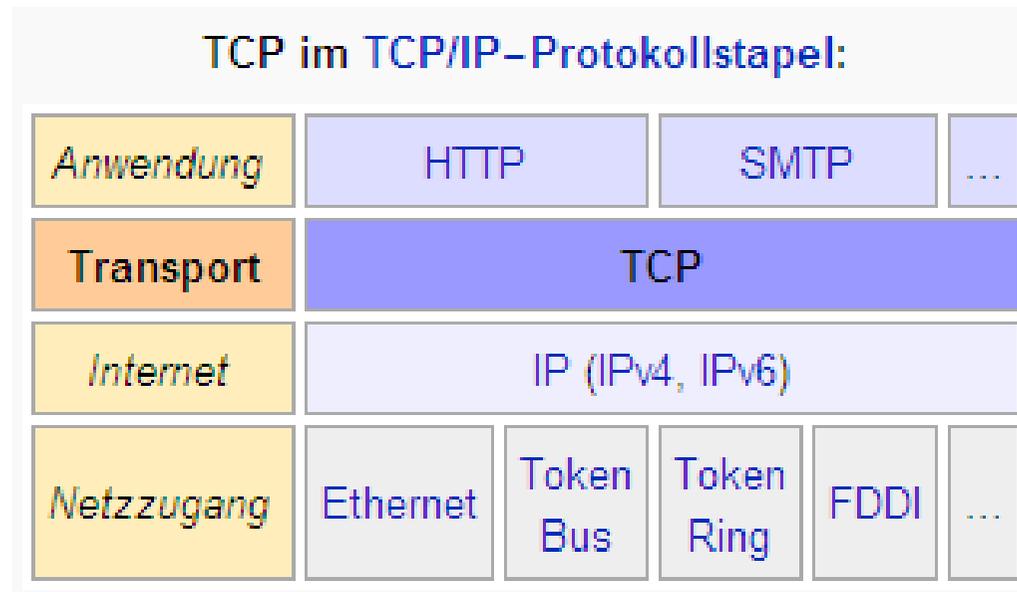




# TCP Transmission Control Protocol

---

- o Zuverlässiger, bidirektionaler Datentransport
- o RFC 793 (1981), RFC 1323 (1992)



Quelle: [http://de.wikipedia.org/wiki/Transmission\\_Control\\_Protocol](http://de.wikipedia.org/wiki/Transmission_Control_Protocol)



# TCP Transmission Control Protocol

---

## TCP – Ende-zu-Ende Beziehung

- Virtueller Kanal zwischen Sender und Empfänger über Ports
- Socket (Kombination aus Port und IP-Adresse)
  - Port: 16 bit (65 535)
  - Port: 0 ... 32 von der IANA reserviert
  - Portnummer nicht bindend (Vereinbarung zwischen Server und Clients)
  - „Port Scan“
    - Setzt auf IP (Internet Protocol) auf
    - Transportmedium für www, Webbrowser (Port: 80), Web Server
    - Vollduplex
    - TCP Software: Winsock.dll, wsock32.dll, bei Linux im Kernel



# TCP Header

---

0		15	16		31
Source Port			Destination Port		
Sequence Number					
Acknowledge Number					
Offset	Reserved	Flags		Windows Size	
Checksum			Urgent Pointer		
Options				Padding	

Quelle: [http://de.wikipedia.org/wiki/Transmission\\_Control\\_Protocol](http://de.wikipedia.org/wiki/Transmission_Control_Protocol)



# TCP Header

0		15	16		31
Source Port			Destination Port		
Sequence Number					
Acknowledge Number					
Offset	Reserved	Flags		Windows Size	
Checksum			Urgent Pointer		
Options				Padding	

- o **Source Port (Quellport)**
  - o Enthält die Portnummer der Quelldaten
- o **Destination Port (Zielport)**
  - o Bestimmt den Ziel-Port der Daten. Dieser bleibt für die Dauer der Verbindung gleich.
- o **Sequenz Number**
  - o Gibt beim Verbindungsaufbau eine Zufallszahl als „Initial Sequence Number“ (ISN) an. Das erste Segment enthält so den Wert ISN+1.
- o **Acknowledge Number (Quittierungsnummer)**
  - o Bestätigungsnummer für Empfangsquittungen an den Sender
- o **Offset**
  - o Gibt die Anzahl der 32-Bit-Worte im TCP-Header an. Der Eintrag in diesem Feld ist für die Berechnung des Datenteils relevant.
- o **Reserved**
  - o Für zukünftige Anwendungen reserviert; muss immer auf Null gesetzt werden.



# TCP Header

0		15	16		31
Source Port			Destination Port		
Sequence Number					
Acknowledge Number					
Offset	Reserved	Flags		Windows Size	
Checksum			Urgent Pointer		
Options				Padding	

## o Control Flags

o Enthält eine Reihe von so genannten Ein-Bit-Indikatoren, die zum Aufbau , zur Beendigung und zur Aufrechterhaltung von Verbindungen dienen.

## o Windows Size

o Dient zur Flusskontrolle zwischen Sender und Empfänger. Die Flusskontrolle basiert auf der fortlaufenden Nummerierung der übertragenen Datenpakete.

## o Checksum

o Enthält eine Prüfsumme, die aus dem TCP-Header und einem 96-Bit-Pseudo-Header gebildet wird.

## o Urgent Pointer

o Gibt an, dass die TCP-Segmente Informationen mit großer Dringlichkeit transportieren. Solche Segmente werden durch das URG-Flag gekennzeichnet

## o Options

o Definiert Dienstoptionen, Optionenart und Optionenlänge. Die aktuellen Optionendaten bestimmen die Länge des Feldes.

## o Padding

o Enthält eine variable Bit-Zahl, die sicherstellt, dass der TCP-Header bei Benutzung des Options-Feldes immer im 32-Bit-Format endet.



# Aufbau des Pseudo Headers in TCP

TCP-Pseudoheader (IPv4)

Bit offset	Bits 0–3	4–7	8–15	16–31
0	IP-Absenderadresse			
32	IP-Empfängeradresse			
64	00000000	Protocol 6 (=TCP)		TCP-Länge
96	Quellport			Zielport
128	Sequenznummer			
160	ACK-Nummer			
192	Datenoffset	Reserviert	Flags	Window
224	Prüfsumme			Urgent pointer
256	Options (optional)			
256/288+	Daten			

Quelle: [http://de.wikipedia.org/wiki/Transmission\\_Control\\_Protocol](http://de.wikipedia.org/wiki/Transmission_Control_Protocol)



## Übungsaufgabe 19

---

**Um welche Nummern zur eindeutigen  
Verbindungsidentifikation erweitert das TCP das IP? Durch  
welche Größen wird eine TCP-Verbindung eindeutig  
charakterisiert?**



## Übungsaufgabe 20

---

**Welchen Wert hat das Feld Offset, wenn das Feld Options genutzt wird und somit der TCP-Header eine Länge von 32 Bytes hat?**



## Übungsaufgabe 21

---

**Ein Sender sendet über Ethernet (mit 10 Mbit/s) TCP-Nachrichten, die jeweils eine Größe von 1000 Bytes haben. Es können fünf Nachrichten ohne Quittung gesendet werden. Wie lange darf die Quittungszeit höchstens sein, damit der Sender – falls kein Fehler auftritt – kontinuierlich senden kann? (Darunter soll die Zeit zwischen Senden des letzten Bits und Empfang der zugehörigen Quittung verstanden werden.)**



## Übungsaufgabe 22

---

**Welcher Unterschied besteht zwischen einem TCP und einem UDP Protokoll?**

**Welche Vor- und Nachteile hat ein UDP Protokoll gegenüber einem TCP Protokoll?**



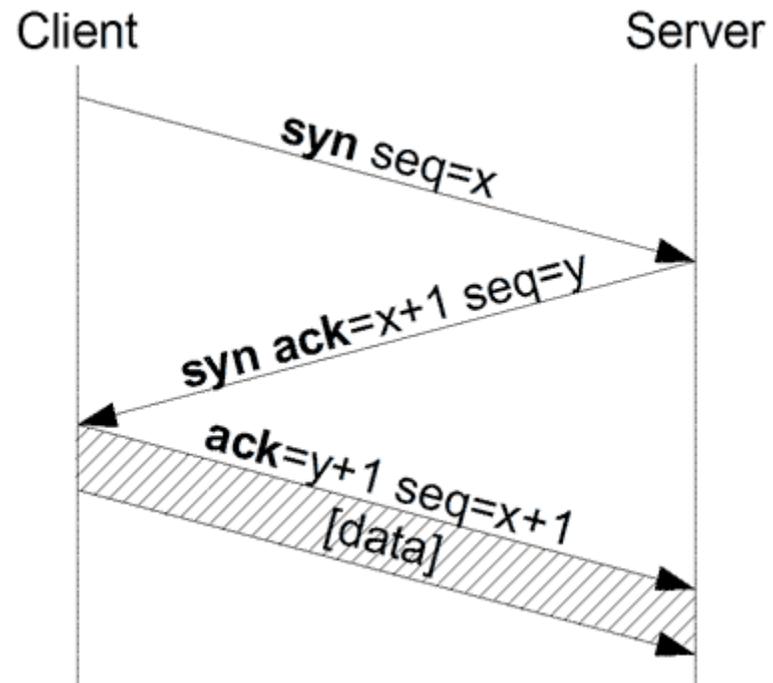
# Übungsaufgabe

---

**Wie funktioniert der Verbindungsaufbau mit Drei Wege  
Handshake unter TCP?**



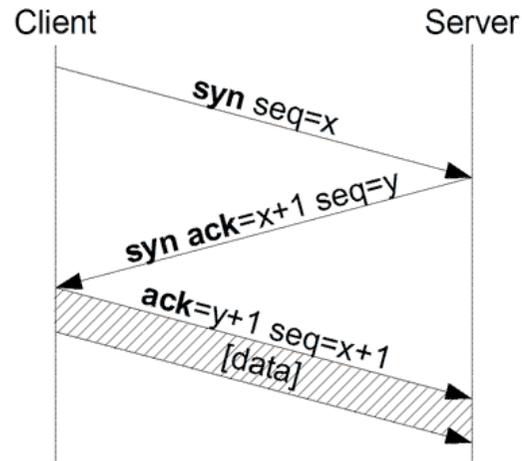
# Drei Wege Handshake - Verbindungsaufbau



Quelle: <http://de.wikipedia.org/wiki/Drei-Wege-Handshake>



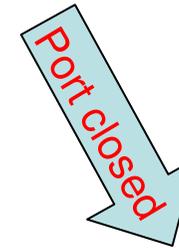
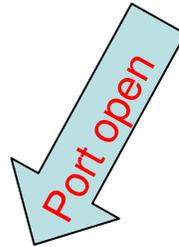
# Drei Wege Handshake - Verbindungsaufbau



Quelle: <http://de.wikipedia.org/wiki/Drei-Wege-Handshake>

## SYN/ACK-Paket

Client sendet dem Server ein SYN-Paket (synchronize) mit einer Sequenznummer  $x$



Server bestätigt das erste SYN-Paket und stimmt dem Verbindungsaufbau mit der Rücksendung eines SYN/ACK-Paket zu. Er bestätigt den Empfang durch die Rückmeldung  $x + 1$  und fügt eine Start Sequenz  $y$  hinzu. Der Client bestätigt wiederum  $y + 1$  und  $x + 1$ .

Die Verbindung ist damit aufgebaut.

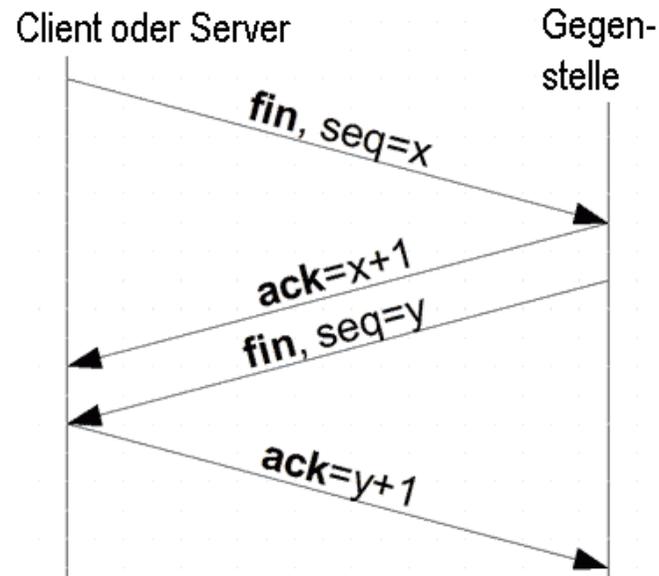
Die Verbindung ist für beiden Kommunikationspartner gleichberechtigt.

## TCP-RST

Server empfängt Paket, Port geschlossen, RCP-RST signalisiert das keine Verbindung aufgebaut werden kann.



## Drei Wege Handshake - Verbindungsabbau



Quelle: <http://de.wikipedia.org/wiki/Drei-Wege-Handshake>

Verbindungsabbau erfolgt ähnlich dem Verbindungsaufbau.

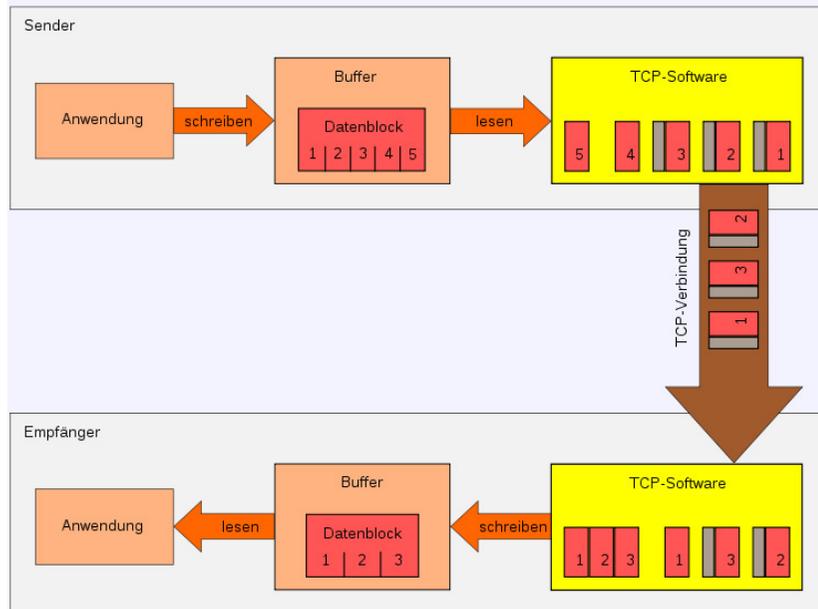
Statt des SYN-Bits kommt das FIN-Bit zum Einsatz

Der Empfänger bestätigt den Verbindungsabbau  $ack = x + 1$  und erzeugt ebenfalls ein FIN-Bit mit der Sequenznr  $y$

Zum Abschluss wird der Verbindungsabbau mit  $ack = y + 1$  bestätigt.



# Nutzdatensegmentierung



Quelle: [http://de.wikipedia.org/wiki/Transmission\\_Control\\_Protocol](http://de.wikipedia.org/wiki/Transmission_Control_Protocol)

## Maximum Segment Size (MSS)

- o Maximale Anzahl von Bytes, die als **Nutzdaten** in einem TCP Segment versendet werden können (Transportschicht).
- o Verbindungsaufbau:
  - o Client teilt Server MSS Wert mit
  - o Server antwortet mit seinem MSS Wert.
  - o Beide einigen sich auf den niedrigeren Wert.

## Maximum Transmission Unit

- o Diese wird von der Netzwerk-Hardware bzw. vom Netzwerk-Protokoll vorgegeben.
- o Sie beschreibt die maximale Paketgröße eines Protokolls der Vermittlungsschicht (OSI, Schicht 3) bzw. der Netzwerkschicht (TCP/IP, Schicht 2), welche ohne eine weitere Fragmentierung übertragen werden kann.



# Nutzdatensegmentierung

Typische MTU-Größen

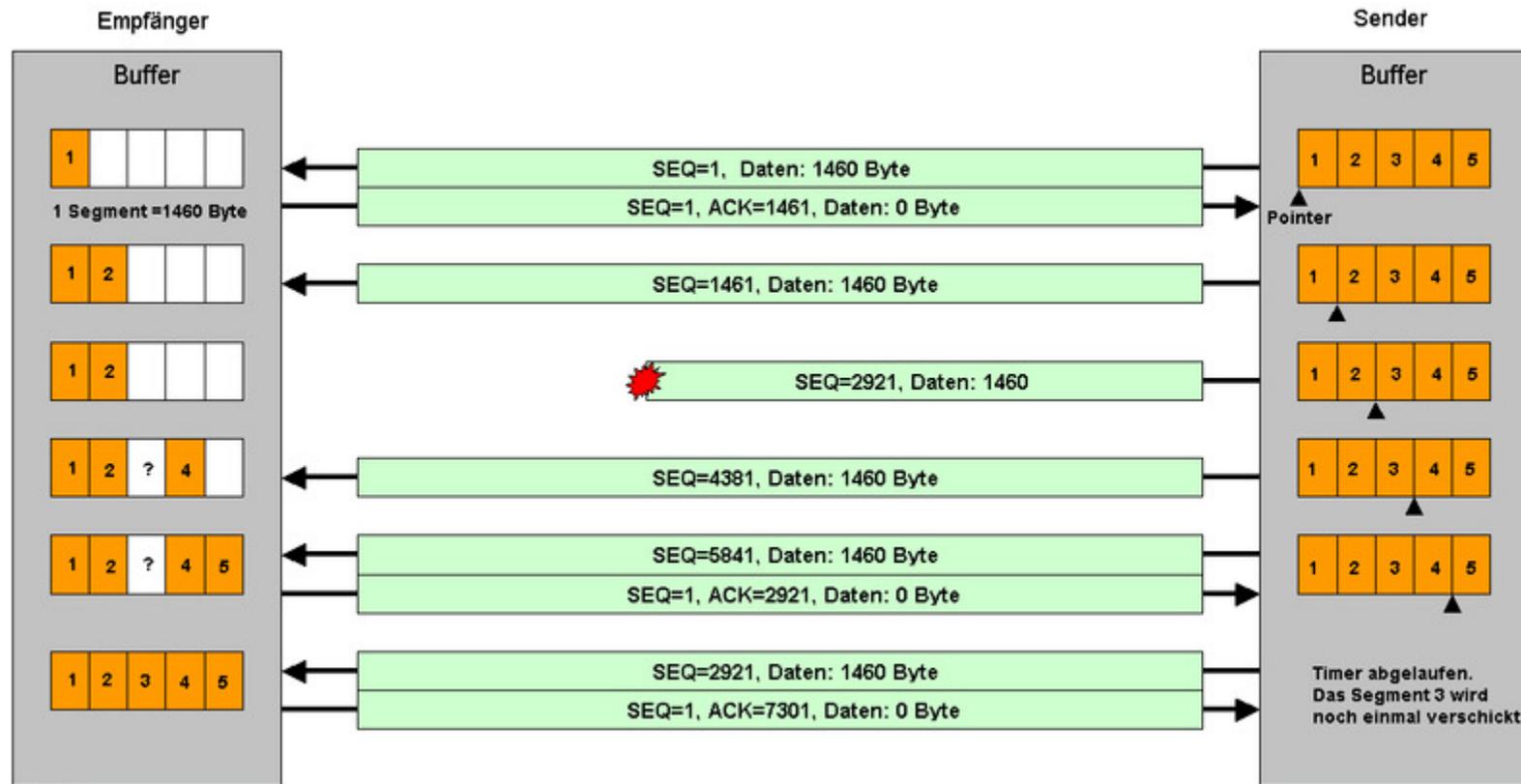
Medium	MTU in Bytes
Hyperchannel	65535
Token Ring(4)(802.5)	4464
Token Ring(16)	17914
FDDI	4352
Ethernet	1500
Gigabit Ethernet mit Jumboframes	9000
PPPoE (z. B. DSL)	≤ 1492
SLIP/PPP (low delay)	296
X.25	576
FibreChannel	theoretisch unbegrenzt
ISDN	576
DQDB	
HIPPI	
ATM	4500, s. u.
ARCNET	
802.11	2312 (WiFi)

Quelle: [http://de.wikipedia.org/wiki/Maximum\\_Transmission\\_Unit](http://de.wikipedia.org/wiki/Maximum_Transmission_Unit)

- o TCP Segment: typischerweise 1500 Bytes (12000 Bit)
- o IP theoretisch bis 65535 Bytes (64 kB)
- o Beispiel: IPv4, IP- und TCP-Header jeweils 20 Bytes  
=> Nutzdaten bei Verwendung des Ethernet Protokolls 1460 Bytes. MSS muss somit mind. 40 Oktetts (Bytes) kleiner sein als die MTU um eine Fragmentierung der IP-Pakete zu vermeiden.



# Beispiel einer TCP/IP-Datenübertragung



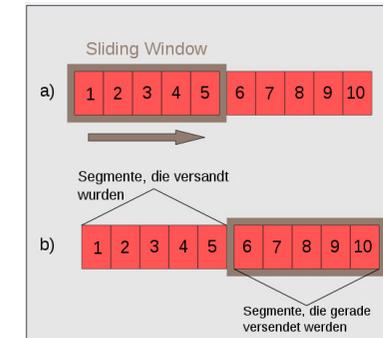
Quelle: [http://de.wikipedia.org/wiki/Transmission\\_Control\\_Protocol](http://de.wikipedia.org/wiki/Transmission_Control_Protocol)



# Flusssteuerung

## Problem:

- o Werden Pakete schneller gesendet, als sie der Empfänger verarbeiten kann, hat dies Konsequenzen.
  - neu ankommende Segmente müssen verworfen werden
  - daraus resultieren Sendewiederholungen, die die Datenübertragung verlangsamen und Sender und Empfänger zusätzlich belasten



Quelle: [http://de.wikipedia.org/wiki/Transmission\\_Control\\_Protocol](http://de.wikipedia.org/wiki/Transmission_Control_Protocol)

## Lösung:

- o Der Empfänger teilt dem Sender durch den Sliding-Window-Mechanismus mit, wie viele Segmente er (noch) aufnehmen kann.

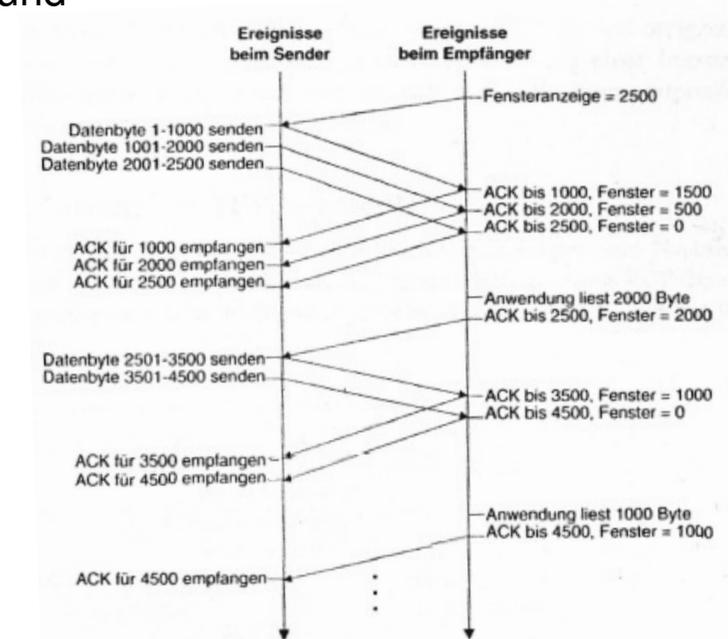


Abb. 4: Beispiel einer TCP-Flusssteuerung

Quelle: Douglas Corner, Computernetzwerke und Internets, 2004