Internet-Transport
TCP – virtuelle Verbindung zwischen Anwendungen

STUDIEREN AUF HÖCHSTEM NIVEAU

Prof. Dr. Jürgen Anders, Hochschule Furtwangen
Fakultät Digitale Medien

Protokolle der Transportschicht

Protokolle der Transportschicht setzen auf dem **verbindungslosen Datagrammdienst** IP der Internetschicht auf

- Prominentestes Protokoll ist TCP – Transmission Control Protocol
- TCP leistet dabei scheinbar Unmögliches:
- Auf Basis des unzuverlässigen IP-Datagrammdienstes bietet TCP einen zuverlässigen, gesicherten Transportdienst zum Datenaustausch zwischen zwei Endsystemen

Grundidee:

- TCP gleicht Fehler des IP-Datagrammdienstes aus …
- Erfolg des Internets ist neben IP vor allem dem glücklichen Design des TCP-Protokolls zu verdanken -> TCP/IP-Protokollsuite

In der TCP/IP-Protokollsuite gibt es **zwei grundlegende Transportprotokolle**:

- TCP – Transmission Control Protocol:
- bietet gesicherten, verbindungsorientierten Vollduplex-Kommunikationsdienst im Internet
- ist sehr komplex und unterliegt steter Weiterentwicklung
- UDP – User Datagram Protocol:
- bietet sehr einfachen, ungesicherten, verbindungslosen Kommunikationsdienst im Internet
- aufgrund seiner Einfachheit bisher kaum Veränderungen unterworfen

Transportschicht im TCP/IP Schichtenmodell: TCP/IP-Stack

TCP/IP-Stack

5	Verarbeitung (Application Layer)
4	**Transport (Transport Layer)**
3	Vermittlung (Network Layer)
2	Sicherung (Link Layer)
1	Bitübertragung (Hardware Layer)

TCP Ports

Um TCP-Verbindung aufrechterhalten zu können, müssen Sender und Empfänger jeweils Endpunkte – **Sockets** – für die Kommunikation einrichten

- Jeder Socket verfügt über reservierten Speicherplatz zum Puffern der empfangenen oder zu sendenden Daten
- Zuordnung der Sockets ist auf beide Kommunikationspartner und Zeitdauer der jeweiligen TCP-Verbindung beschränkt
- Jeder Socket besitzt **Socket-Nummer**, die sich aus
  - IP-Adresse des Rechners und
  - lokal zuordenbarer 16-Bit-Nummer – Port – ergibt
- Ports sind Service Access Points der Transportschicht
- Ports mit Nummern
  - 0 – 1.023 – **Well Known Ports** – sind weltweit eindeutig für Standarddienste reserviert, z.B. 80 – HTTP, 23 – Telnet, 53 – DNS, …
  - 1.024 – 65.535 können beliebigen Anwendungsprogrammen zugeordnet werden

Man unterscheidet:

- Reservierte Ports:
  - 0 – 255 für TCP/IP-Anwendungen
  - 256 – 1.023 für bestimme UNIX-Anwendungen
- Registrierte Ports:
  - 1.024 – 49.151 von IANA registriert
- Private, dynamische Ports: Rest

Portnummern sind zu wählen beim Verbindungsaufbau zwischen zwei Anwendungsprogrammen:

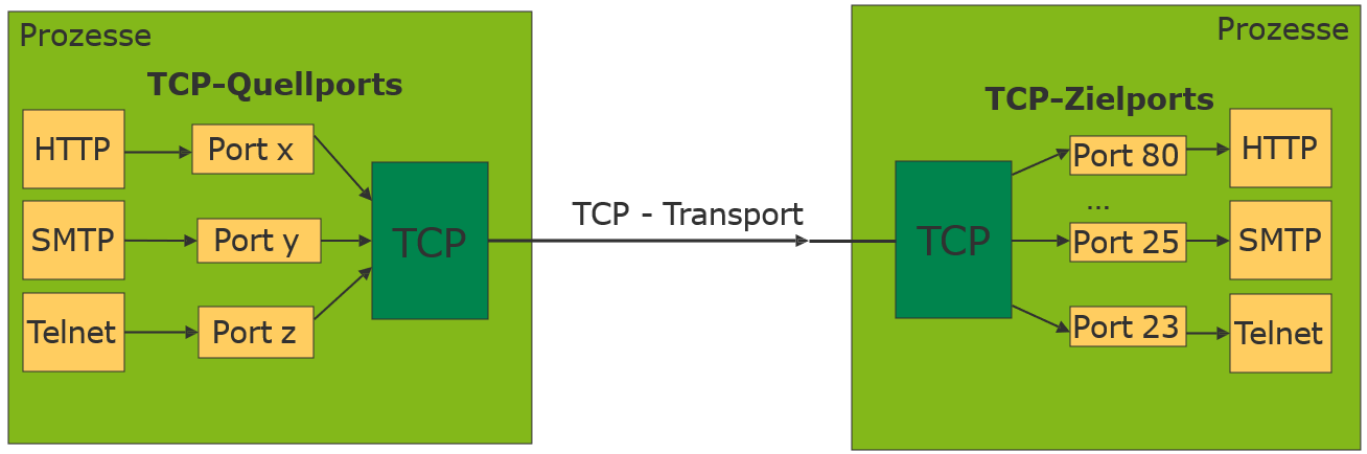
- Zielport: Portnummer der gewünschten Anwendung
- Quellport: Nicht reservierte und zur Zeit nicht anders genutzte Portnummer

TCP-Verbindung kann anhand der beiden Socket-Nummern über das Tupel (Socket Sender, Socket Empfänger) eindeutig identifiziert werden

- Ein Server-Socket kann gleichzeitig mehreren Verbindungen dienen

Anwendungsprogramme nutzen den TCP-Transportdienst über sogenannte TCP-Primitive – Primitives, z.B. Request, Response, Confirm, Read, Write, …, die über den Socket implementiert sind

TCP–Verbindungen zwischen verschiedenen Anwendungsprozessen verlaufen über TCP-Ports



Virtuelle Verbindung mit TCP

In Rechnernetzen und Zusammenschlüssen gibt es **zwei Typen** von Diensten:

Verbindungslose Dienste:

- Kommunikation erfolgt ohne vorherigen Verbindungsaufbau
- Pakete werden einfach in das Netzwerk abgeschickt -> Beispiel: Datagrammdienste IPv4, IPv6

Verbindungsorientierte Dienste:

- Kommunikation erfolgt erst nach Aufbau einer Verbindung
- alle Datenpakete werden demzufolge in korrekter Reihenfolge versendet und empfangen -> Beispiel: TCP

TCP bietet verbindungsorientierten Dienst

- TCP stellt aber lediglich virtuelle Verbindung bereit, d.h. TCP-Prozesse auf den Endsystemen täuschen Verbindung softwaretechnisch vor
- Wie schafft TCP die Illusion der Verbindung?
  - Mit Hilfe sogenannter Sequenznummern wird jeweils jede von A nach B und jede von B nach A gesendete TCP-Nachricht durchnummeriert
  - Anhand der Sequenznummern kann TCP beim Empfänger Vollständigkeit, richtige Reihenfolge, Duplikatfreiheit, … der gesendeten TCP-Nachrichten rekonstruieren
- Vor eigentlicher Datenübertragung baut TCP virtuelle Verbindung auf zum designierten Empfänger, die nach vollzogener Daten-Übertragung wieder abgebaut wird
  - Hierbei müssen insbesondere in beide Richtungen die Sequenznummern vereinbart und bestätigt werden
- TCP muss nur in den Endsystemen implementiert sein, nicht in den Zwischensystemen
  - TCP nutzt IP-Datagrammdienst zur Datenübertragung
  - jede TCP-Nachricht wird gekapselt in IP-Datagrammen über das Internet übertragen
- IP behandelt TCP-Nachrichten als reine Nutzdaten

Virtuelle Verbindung mit TCP

Ende-zu-Ende-Übertragung

TCP erlaubt ausschließlich Datenübertragungen zwischen zwei dezidierten Endsystemen

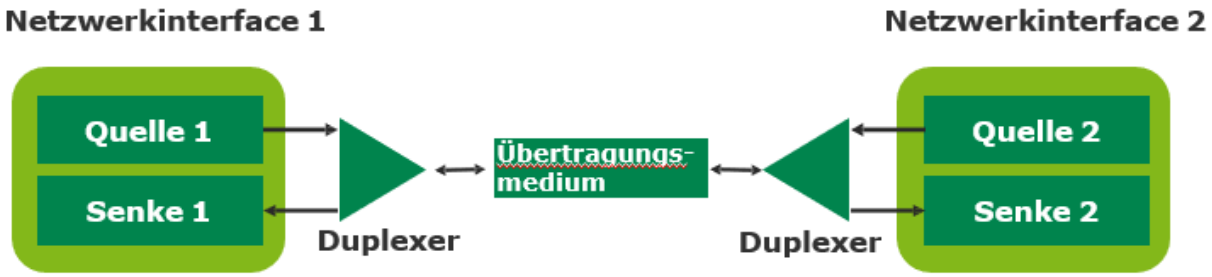
- Verbindung verläuft von einer Anwendung beim Sender zu einer Anwendung beim Empfänger
- Aufbau der virtuellen Verbindung ist allein Sache der beiden Endsysteme, die zu überbrückenden Netze und Zwischensysteme leiten Nachrichten nur in Form von IP-Datagrammen weiter
- Multicast oder Broadcast ist mit TCP nicht möglich

TCP garantiert

- fehlerfreie Datenübertragung,
- kein Datenverlust,
- keine vertauschten Datenpakete,

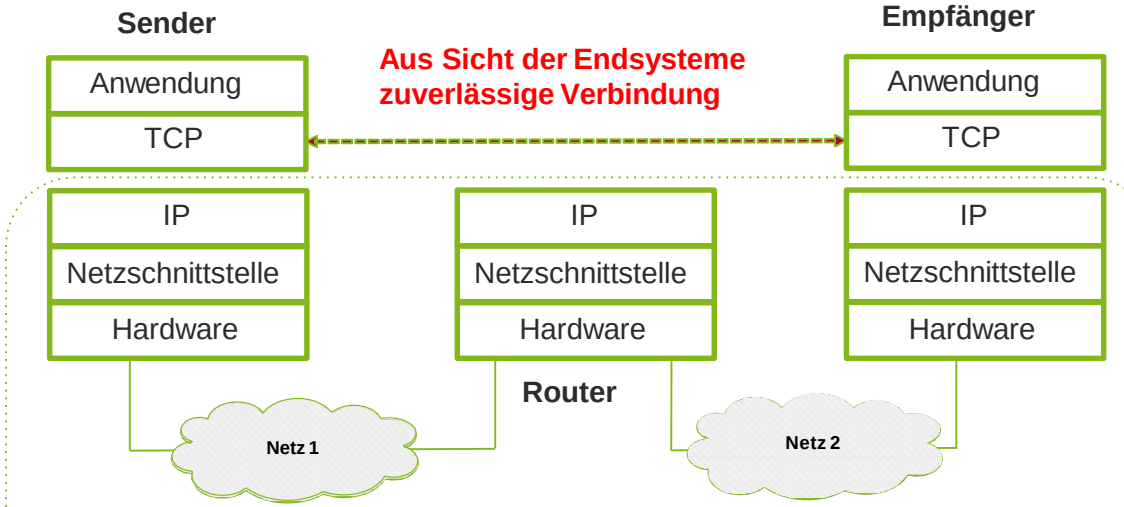
TCP bedient sich dazu der folgenden Techniken

- Retransmission – Neuübertragung
- Adaptive Neuübertragung
- Überlastkontrolle – Congestion Control



Vollduplex Übertragung

- TCP gestattet bidirektionale Ende-zu-Ende Datenübertragung
- Jeweils ausgestattet mit Eingangspuffern können Sender und Empfänger gleichzeitig senden



Tatsächlich verfügen Netze und Zwischensysteme nicht über TCP

TCP Verbindungsauf- und abbau

Zuverlässiger Verbindungsauf- und Abbau

- TCP verlangt, dass Sender und Empfänger vor ihrer Datenübertragung eine Verbindung aufbauen
- eventuelle Nachrichtenduplikate aus früheren Verbindungen können so ignoriert werden
- Wollen Kommunikationspartner die Verbindung beenden, sorgt TCP vor Abbau der Verbindung dafür, dass alle gesendeten, aber noch nicht angekommenen TCP-Nachrichten zugestellt werden
- Zum Auf- bzw. Abbau von TCP-Verbindungen werden drei bzw. vier Nachrichten ausgetauscht -> Drei-Wege-Handshake

Drei-Wege-Handshake sorgt dafür, dass beide Seiten zum Datenempfang bereit sind und die zur Identifikation der Verbindung essentiellen Sequenznummern ausgetauscht werden

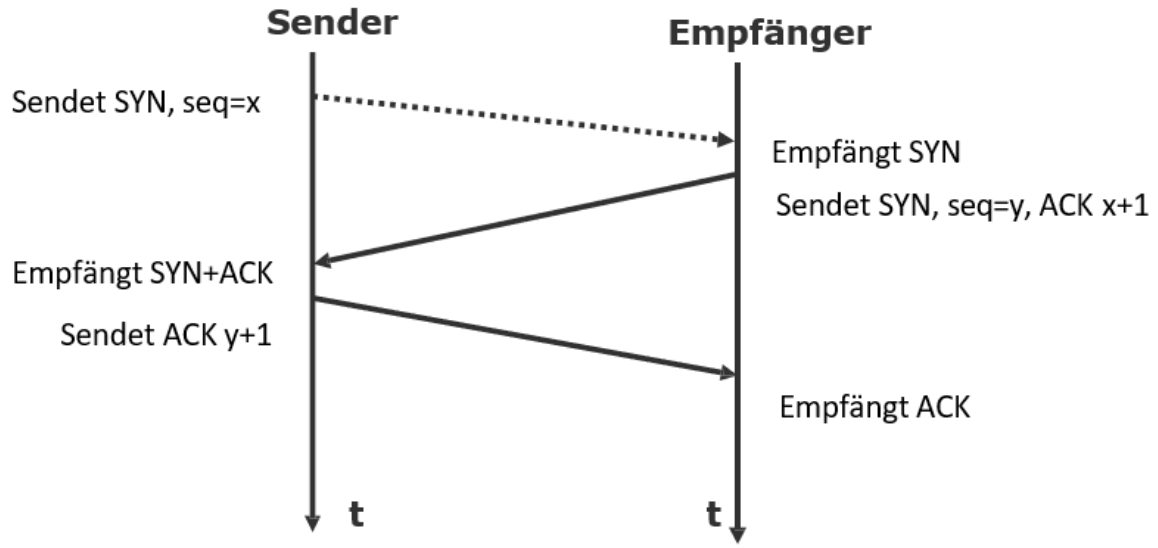
Drei-Wege-Handshake zum Verbindungsaufbau:

- Initiator sendet spezielles Segment - SYN-Segment - , das eine Sequenznummer x enthält und dessen Synchronisationsbit auf 1 gesetzt ist
- Empfänger des SYN-Segments bestätigt die im SYN-Segment enthaltene Sequenznummer x+1, lässt Synchronisationsbit gesetzt und sendet eigene Sequenznummer y
- Initiator bestätigt Empfang der Synchronisations-Antwort mit Sequenznummer y+1

Drei-Wege-Handshake zum Verbindungsabbau:

- Initiator (Sender) sendet spezielles Ende-Segment – FIN-Segment, dessen FIN-Bit auf 1 gesetzt ist und das eine End-Sequenznummer x enthält
- Gegenseite bestätigt übermittelte End-Sequenznummer und nimmt für diese Verbindung keine weiteren Segmente mehr an. TCP-Instanz benachrichtigt lokale Anwendung über Verbindungsabbau
- Informierte Anwendung startet ihrerseits Verbindungsabbau durch Versand eines FIN-Segments mit eigener End-Sequenznummer y und erneuter Bestätigung der zuletzt empfangen Sequenznummer
- Sender bestätigt Empfang und die übermittelte End-Sequenznummer

Verbindungsaufbau mit dem Drei–Wege–Handshake:



TCP Segmente

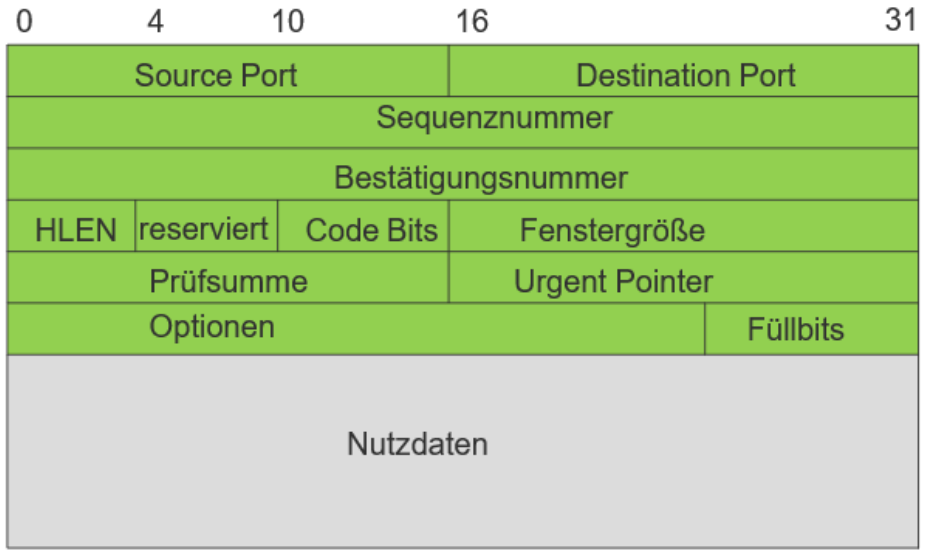
TCP überträgt Nachrichten in Form von einzelnen Datenblöcken – TCP-Segmente

- Bei Aufteilung der Nachricht in Segmente – **Segmentierung** – wird jedes Segment mit eigenem TCP-Header versehen, der alle notwendigen Steuer- und Kontrollinformationen enthält
- Maximale Segmentgröße wird jeweils beim Verbindungsaufbau vereinbart
- Zum Transport werden TCP-Segmente in IP-Datagrammen gekapselt und mit dem IP-Datagrammdienst gesendet
- Geht IP-Datagramm mit TCP-Segment verloren, wird das
- TCP-Segment nicht bestätigt und nach Timeout neu übertragen

Aufbau des TCP-Headers:

- **Source Port** (16 Bit) – Quell-Port, der mit Anwendungsprozess assoziiert ist, der die Verbindung gestartet hat
- **Destination Port** (16 Bit) – Ziel-Port, ist mit Anwendungsprozess assoziiert, der die übertragenen Daten entgegennimmt
- **Sequenznummer** (32 Bit)
  - Zur Durchnummerierung der in Senderichtung übermittelten Segmente
  - Initialer Wert wird beim Verbindungsaufbau ausgetauscht
  - Erhöhung erfolgt jeweils um Anzahl der bereits gesendeten Bytes
- **Bestätigungsnummer** (32 Bit) – dient auf Empfängerseite zur Bestätigung empfangener Segmente
- **Header Length** (4 Bit) – Länge des TCP-Headers in 32-Bit Worten
- **Code Bits** (6 Bit) – zeigen an, welche Felder im Header gültig sind
- **Sendefenster – Window** (16 Bit) – Zahl der Bytes, die Empfänger in seinen Eingabepuffer aufnehmen kann
- **Prüfsumme** (16 Bit) – berechnet aus TCP-Header, den TCP- Nutzdaten und einem Pseudo-Header (der aus TCP- und IP-Daten berechnet wird)
- **Urgent Pointer** (16 Bit) – zeigt an, dass neben den Nutzdaten dringliche – Out-of-Band – Daten, z.B. Interrupts, übertragen werden, die so schnell wie möglich, z.B. ohne Ablage im Eingangspuffer, an Anwendungsprozess übermittelt werden sollen
- **Optionen** – zur Einbeziehung zusätzlicher Funktionalität, z.B. Maximum Segment Size – MSS, Window Scale Option – WSopt, Timestamps Option – TSopt, Selective Acknowledgement, Connection Count – CC
- **Füllbits** – füllen Länge des TCP-Segment-Headers auf Wortgrenze von 32 Bit auf

Format eines TCP-Segments



Internet-Transport
TCP – virtuelle Verbindung zwischen Anwendungen

STUDIEREN AUF HÖCHSTEM NIVEAU

Prof. Dr. Jürgen Anders, Hochschule Furtwangen
Fakultät Digitale Medien