Datenkommunikation

Transportschicht
TCP (2) und Vergleich mit UDP

Wintersemester 2011/2012

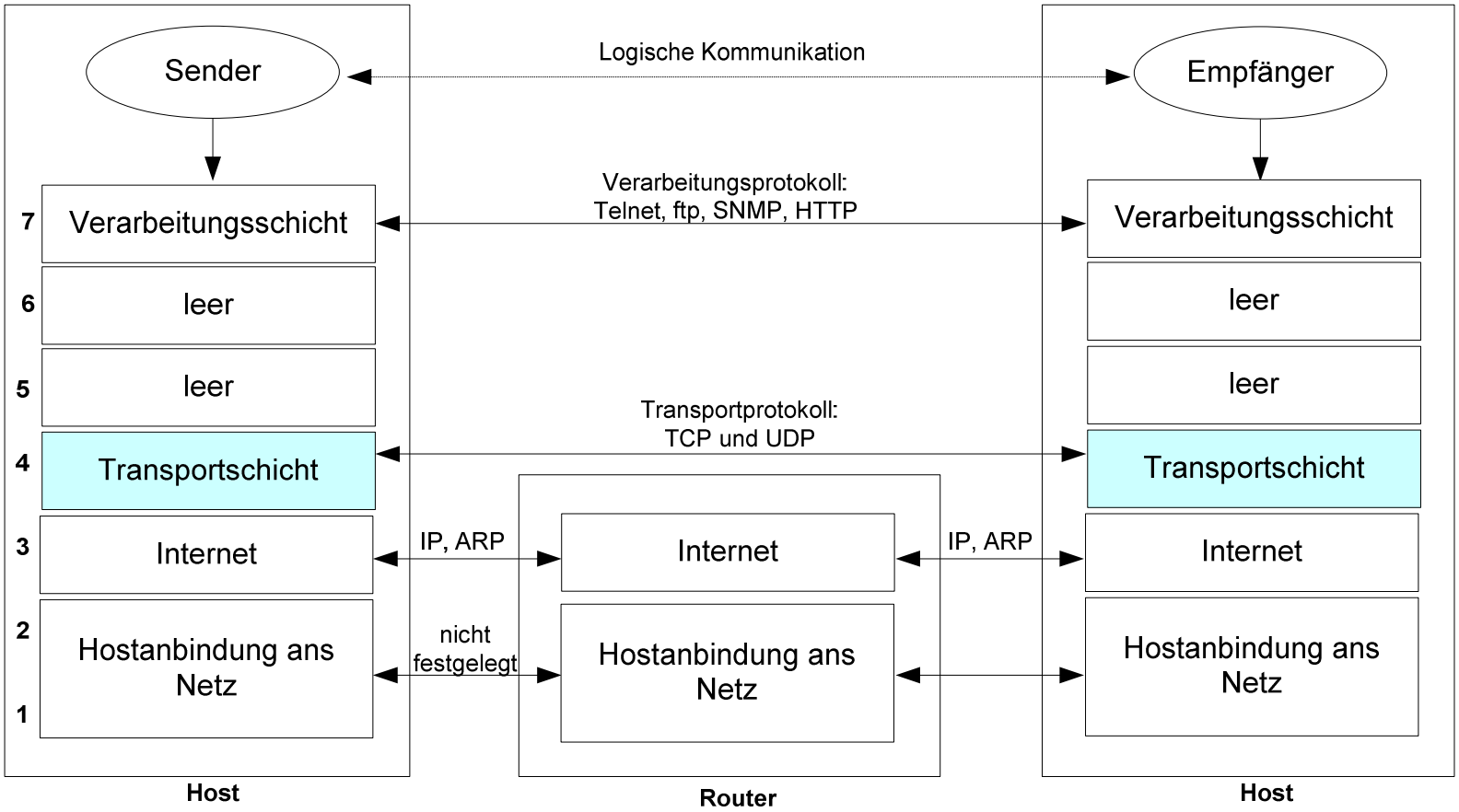
Einordnung

1	Grundlagen von Rechnernetzen, Teil 1
2	Grundlagen von Rechnernetzen, Teil 2
3	Transportzugriff
4	Transportschicht, Grundlagen
5	Transportschicht, TCP (1)
6	Transportschicht, TCP (2) und UDP
7	Vermittlungsschicht, Grundlagen
8	Vermittlungsschicht, Internet
9	Vermittlungsschicht, Routing
10	Vermittlungsschicht, Steuerprotokolle und IPv6
11	Anwendungsschicht, Fallstudien
12	Mobile IP und TCP

Überblick

1. **TCP (Transmission Control Protocol)**
   - **Sliding-Window-Mechanismus**
   - Optimierungen: Algorithmen von Nagle und Clark, Silly Window Syndrom
   - Staukontrolle
   - TCP-Timer
   - TCP-Zustandsautomat
2. UDP (User Data Protocol)
   - Einordnung und Aufgaben des Protokolls
   - Der UDP-Header
   - Datenübertragung

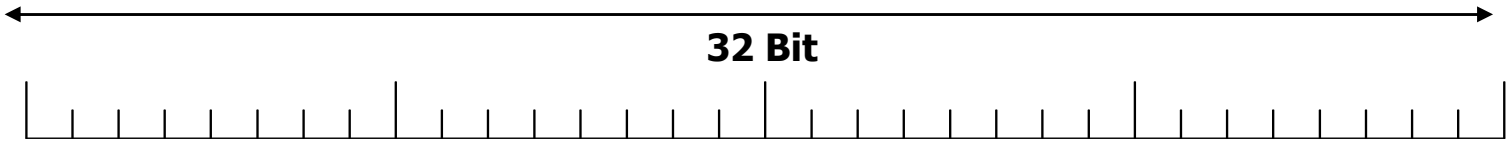
Wiederholung: TCP/IP-Referenzmodell



Sliding Window Mechanismus

- Der Sliding Window Mechanismus erlaubt die Übertragung von mehreren TCP-Sequenzen, bevor ein **ACK**nowlegde eintrifft
- Bei TCP funktioniert Sliding Window auf Basis von Octets (Bytes)
- Die Octets (Bytes) eines Streams sind sequenziell nummeriert
- Flusskontrolle wird über das durch den Empfänger veranlasste Ausbremsen der Übertragung erreicht

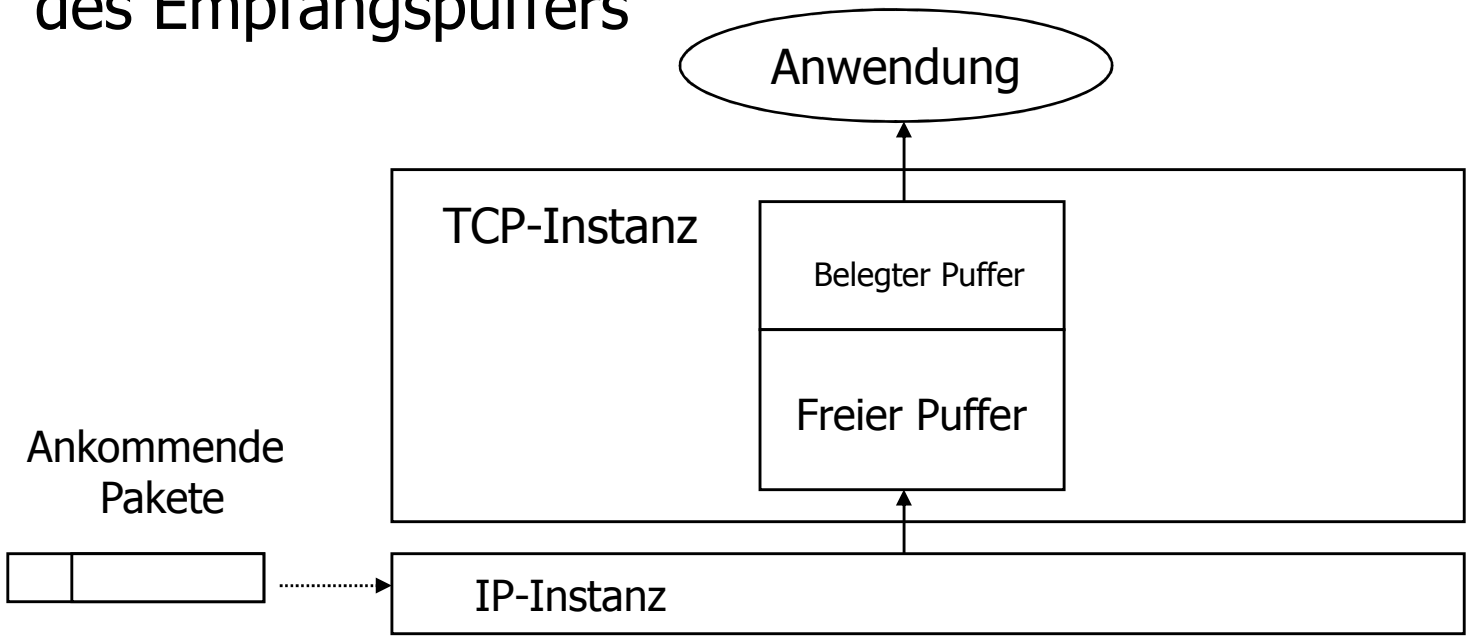
TCP-Header (PCI, Protocol Control Information)



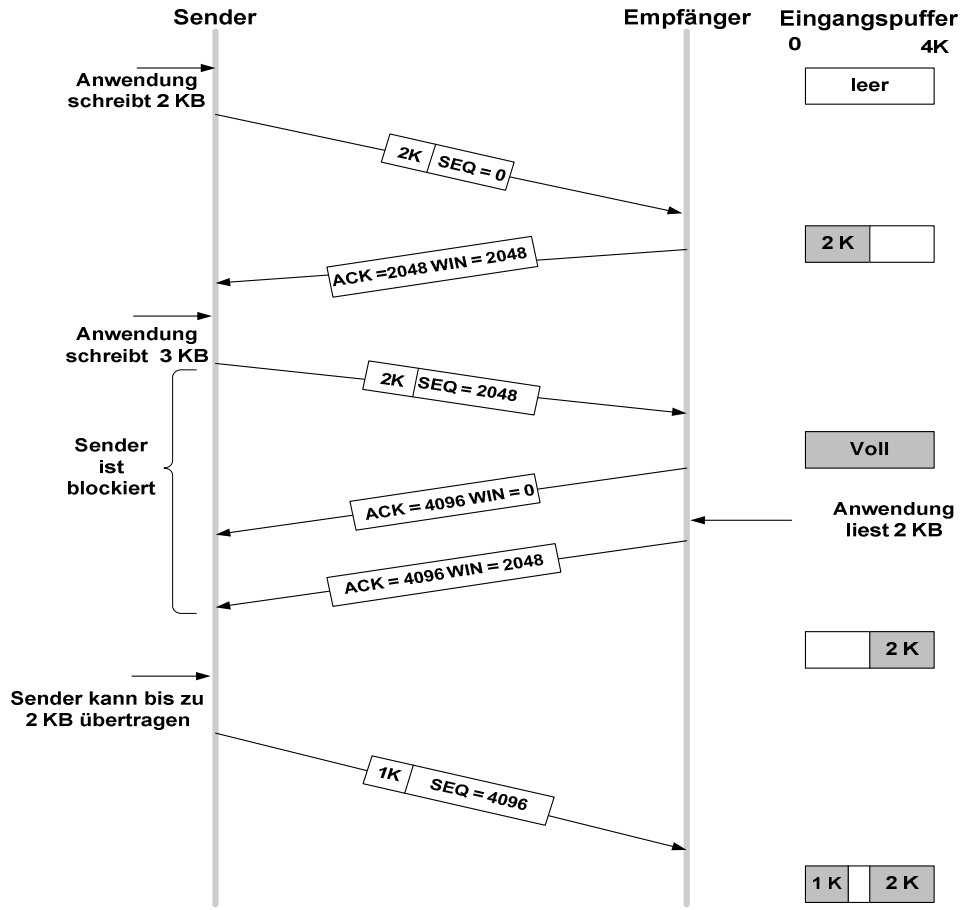
Quellport			Zielport
Folgenummer			
Bestätigungsnummer			
Offset	Reserviert	Flags	**Zeitfenstergröße**
Prüfsumme			Urgent-Zeiger
Optionen (0 oder mehrere 32-Bit-Wörter)			Padding
Daten (optional)			

Sliding Window Mechanismus

- Die TCP-Instanzen reservieren beim Verbindungsaufbau Puffer für abgehende und ankommende Daten
- Empfänger informiert den Sender über den Füllstand des Empfangspuffers

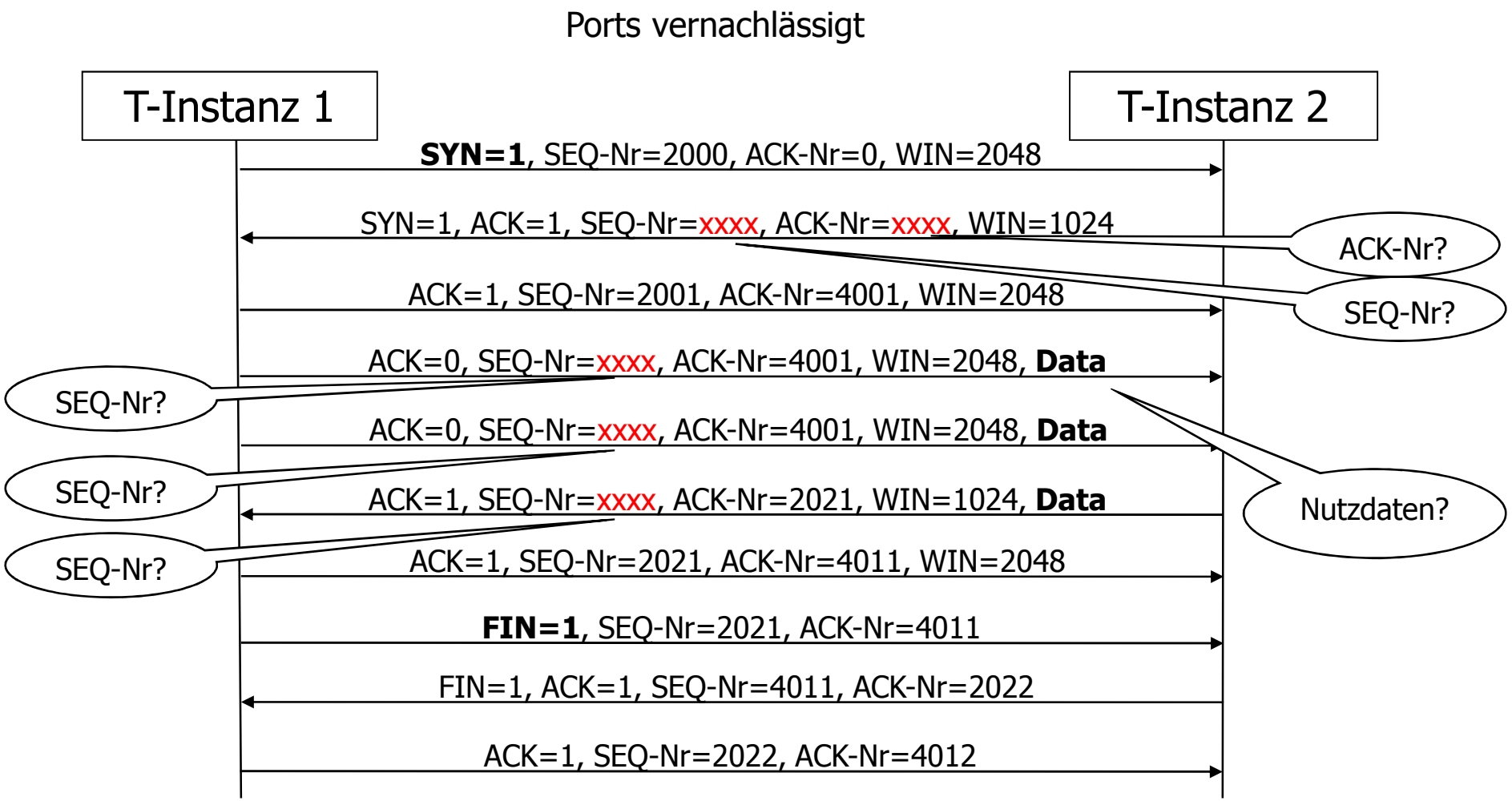


Fenstertechnik, Sliding-Window-Mechanismus

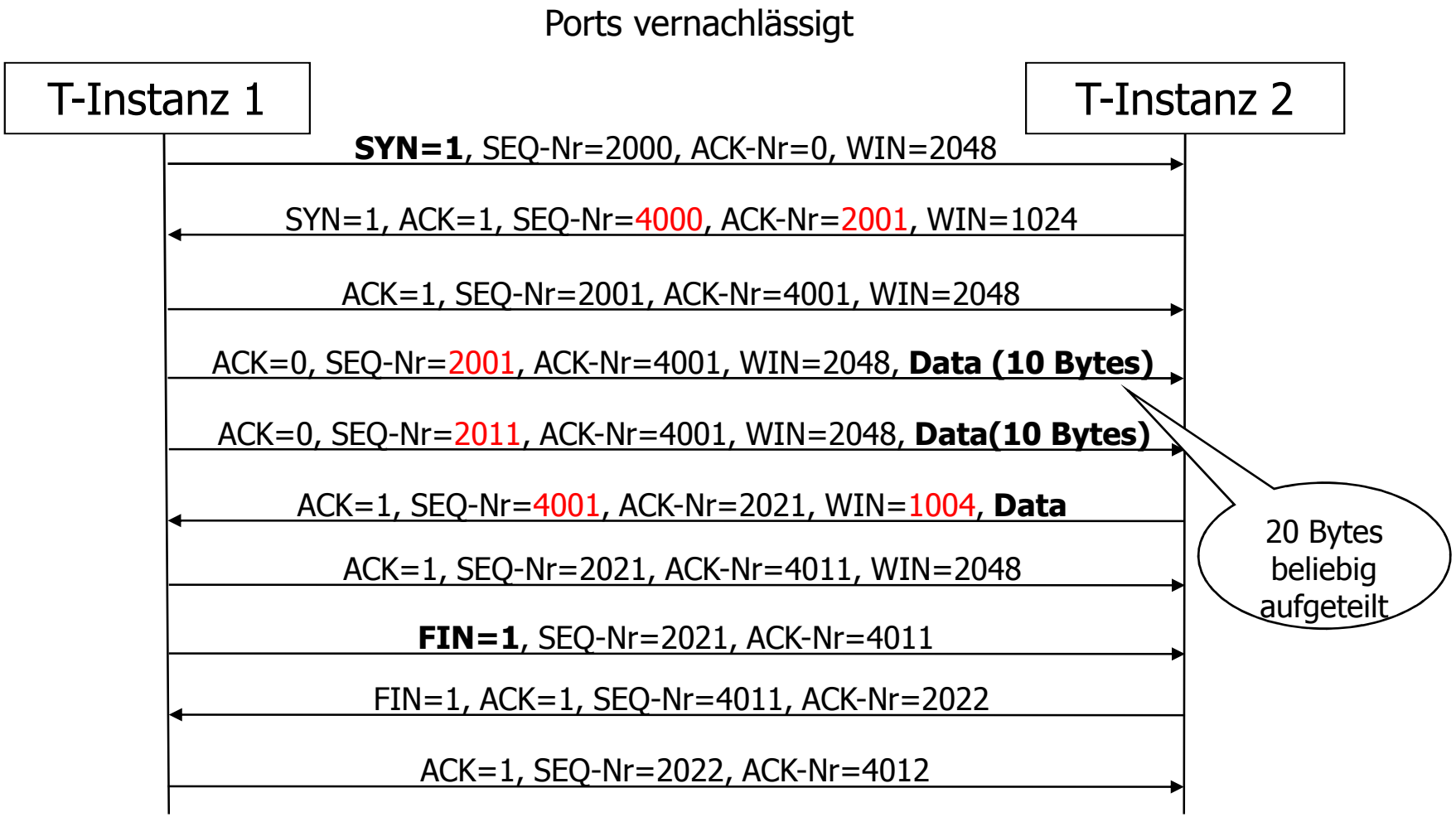


Vgl.Tanenbaum

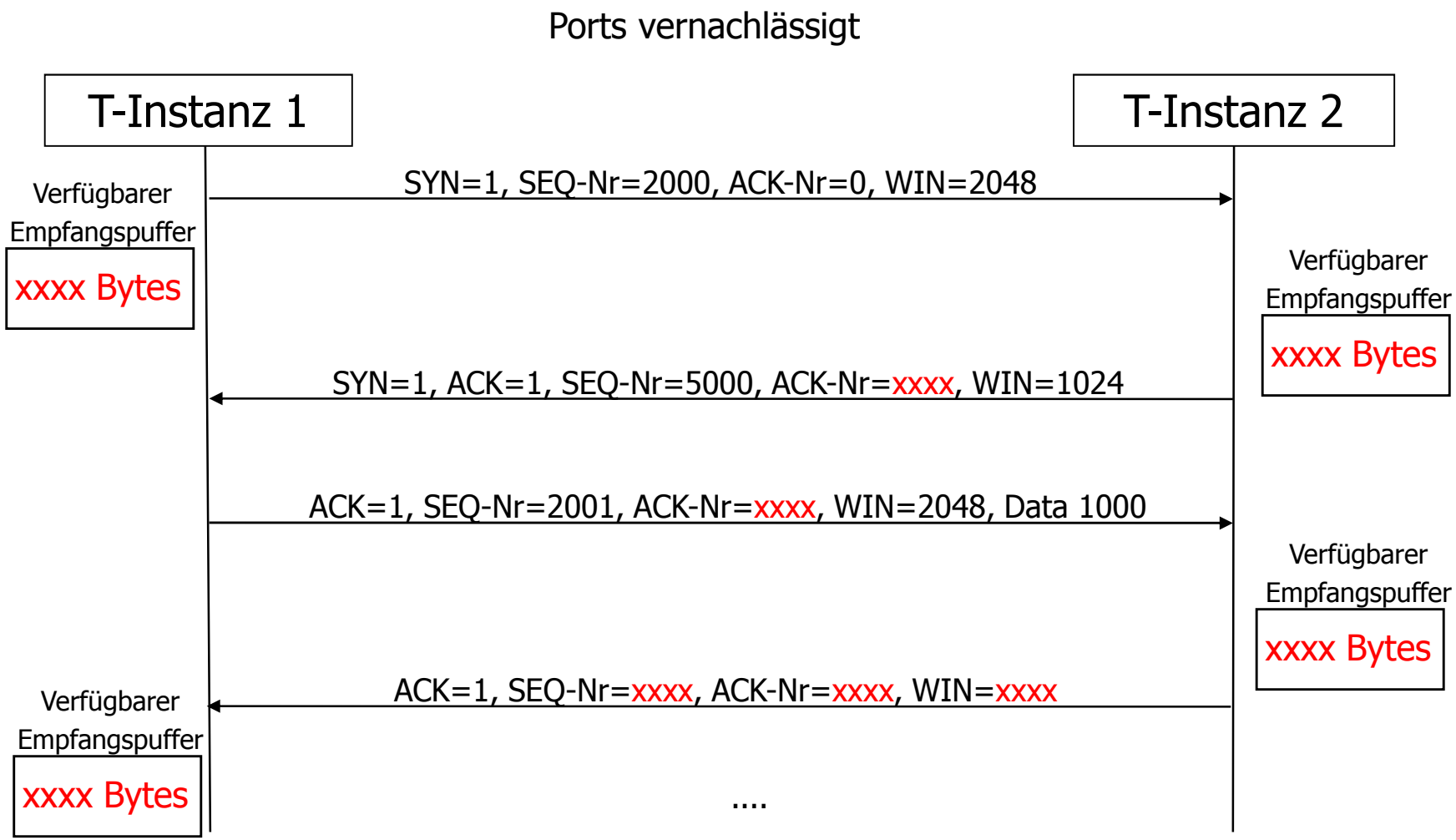
Nachrichtenfluss: Kleine Übung



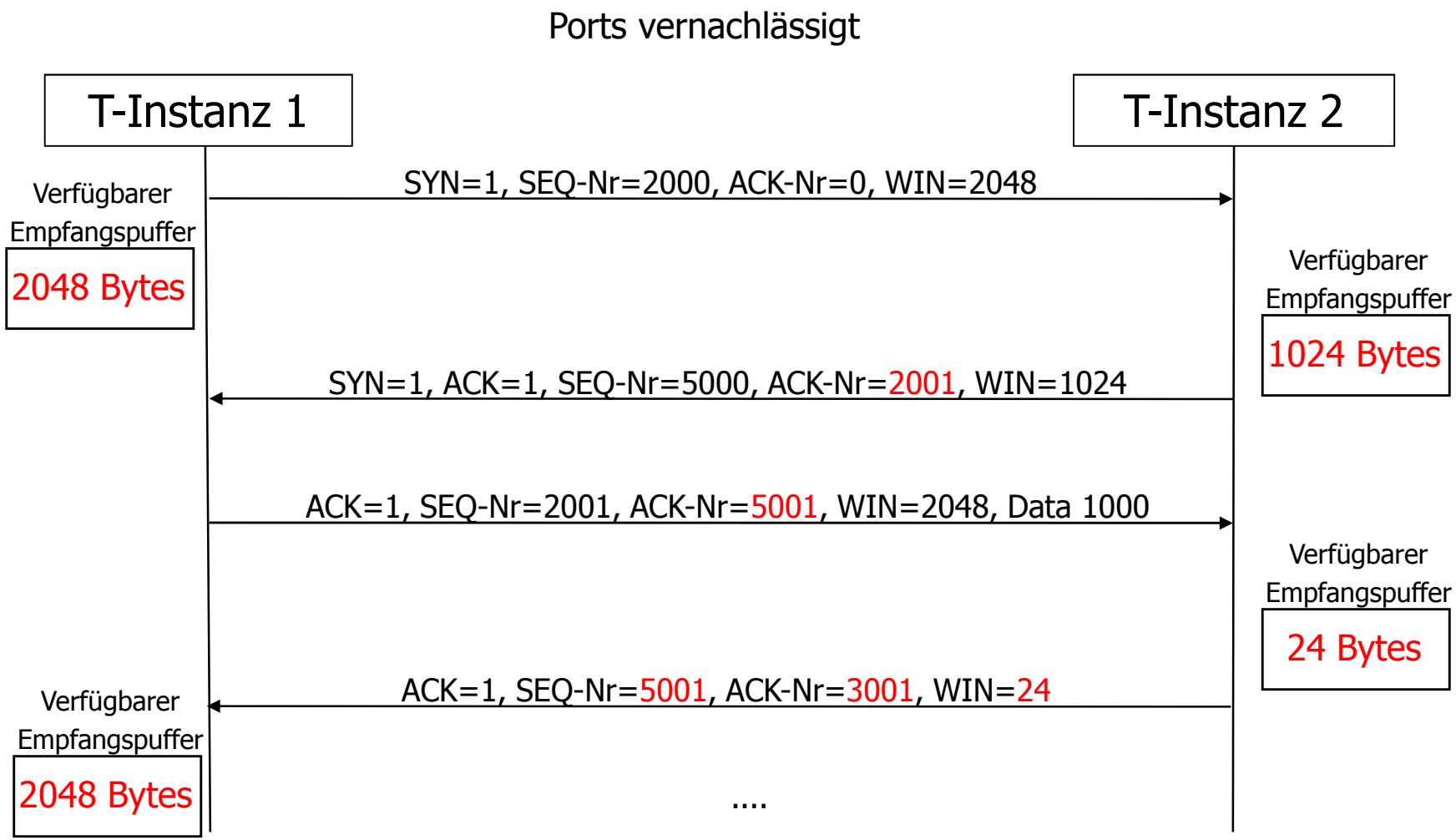
Nachrichtenfluss: Lösung



Nachrichtenfluss: Noch eine Übung



Nachrichtenfluss: Lösung



Überblick

1. **TCP (Transmission Control Protocol)**
   - Sliding-Window-Mechanismus
   - **Optimierungen: Algorithmen von Nagle und Clark, Silly Window Syndrom**
   - Staukontrolle
   - TCP-Timer
   - TCP-Zustandsautomat
2. UDP (User Data Protocol)
   - Einordnung und Aufgaben des Protokolls
   - Der UDP-Header
   - Datenübertragung

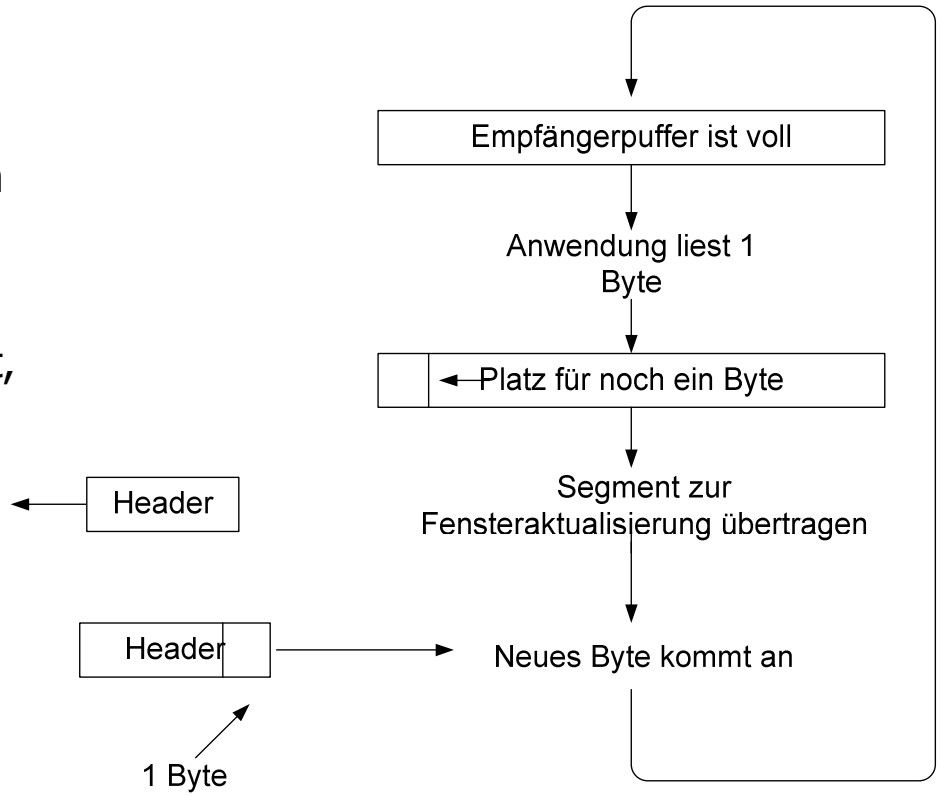
Algorithmus von Nagle

- Optimierung des Sendeverhaltens
  - Nagle versuchte aus Optimierungsgründen zu **verhindern**, dass immer **kleine Nachrichten** gesendet werden
  - Lösungsansatz: Zuerst wird nur 1 Byte gesendet, dann gesammelt und danach erst wieder ein größeres Segment gesendet
  - Kritik: Schlecht bei X-Windows oder bei telnet oder bei ssh. Warum?
  - Daher: Ausschaltbar über Socket-Option (NO_DELAY)
  - Spezifikation siehe RFC 1122

Silly-Window-Syndrom und Algorithmus von Clark

- Optimierung des Bestätigungsverhaltens
  - Problem: Silly Window Syndrom
- Clarks Lösung
  - Verhindert, dass Sende-Instanz ständig kleine Segmente sendet, da Empfängerprozess sehr langsam ausliest
  - Verzögerung der ACK-PDU oder Senden von ACK-PDU mit WIN=0 *)
- Nagle und Clark ergänzen sich in einer TCP-Implementierung

*) Hinweis: Verzögerung bis Empfänger eine halbe Segmentgröße gelesen hat oder der Sendepuffer halb leer ist



Vgl. Tanenbaum

Überblick

1. **TCP (Transmission Control Protocol)**
   - Sliding-Window-Mechanismus
   - Optimierungen: Algorithmen von Nagle und Clark, Silly Window Syndrom
   - **Staukontrolle**
   - TCP-Timer
   - TCP-Zustandsautomat
2. UDP (User Data Protocol)
   - Einordnung und Aufgaben des Protokolls
   - Der UDP-Header
   - Datenübertragung

Staukontrolle bzw. Überlastkontrolle

- **1986** gab es im Internet massive Stausituationen
- Seit **1989** ist Staukontrolle ein wichtiger Bestandteil von TCP
  - J. Nagle: Congestion Control in IP/TCP Internetworks, in RFC 896, 1984
- **IP reagiert nicht** auf Überlastsituationen
- Paketverlust wird von TCP als Auswirkung einer **Stausituation** im Netz interpretiert
- Annahme: Netze sind prinzipiell stabil, **ein fehlendes ACK nach dem Senden einer Nachricht wird als Stau** im Netz **betrachtet**

Staukontrolle bzw. Überlastkontrolle

- TCP **tastet** sich an die maximale Datenübertragungs-rate einer Verbindung **heran**
  - Dies ist die Größe des Überlastfensters
- Das verwendete Verfahren ist das reaktive **Slow-Start**-Verfahren (RFC 1122)
- Es baut auf dem Erkennen von Datenverlusten auf
- Die **Übertragungsrate** wird bei diesem Verfahren im Überlastfall vom Sender massiv **gedrosselt**, die Datenmengen werden kontrolliert
- TCP-Implementierungen **müssen** das Slow-Start-Verfahren unterstützen

Staukontrolle bzw. Überlastkontrolle

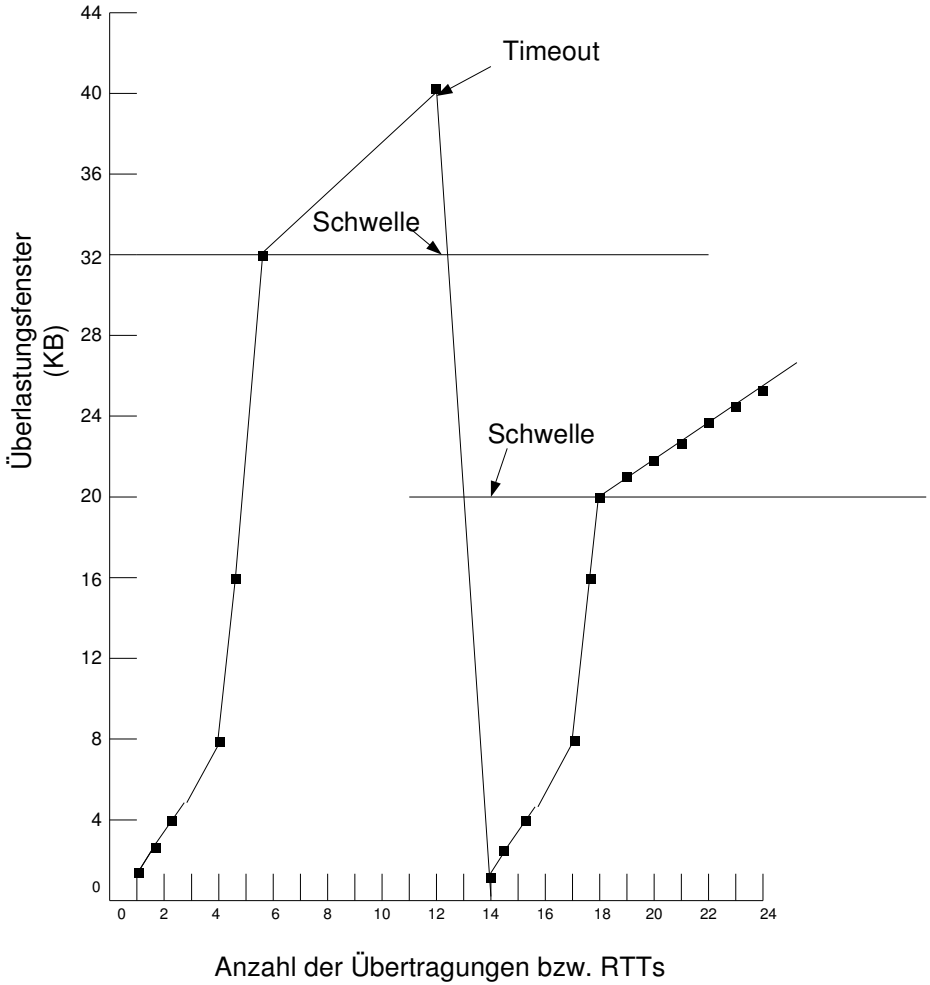
- Es gibt verschiedene Varianten des Slow-Start-Algorithmus
- **Quittungen** dienen als **Taktgeber** für den Sender
- Neben dem Empfangsfenster wird ein neues Fenster eingeführt: das **Staukontrollfenster** (bzw. Überlastungsfenster)
- Es gilt:

  Sendekredit für eine TCP-Verbindung =

  min {Überlastungsfenster, Empfangsfenster}
- **Zwei Phasen**:
  - Slow-Start-Phase
  - Probing-Phase

Staukontrolle bzw. Überlastkontrolle

- Erste TCP-Segmentlänge im Beispiel 1 KB (ausgehandelt)
- 1. Schwellwert bei 32 KB
- Timeout bei Segmentlänge von 40 KB
- Schwellwert wird dann auf 20 KB gesetzt



Staukontrolle bzw. Überlastkontrolle

- Slow-Start-Phase:
  - Sender und Empfänger einigen sich auf eine erste sendbare TCP-Segmentlänge (z.B. 1024 Bytes)
  - Sender sendet Segment dieser Länge
  - Jeweils Verdoppelung der Anzahl an Segmenten bei erfolgreicher Übertragung (exponentielle Steigerung)
  - Ein Schwellenwert (Threshold) wird ermittelt
  - Bei Erreichen des Schwellwerts geht es in die Probing-Phase über

→ Gar nicht so langsam!

Staukontrolle bzw. Überlastkontrolle

- Die Probing-Phase:
  - Bei jeder empfangenen Quittung wird die Größe des TCP-Segments erhöht, aber langsamer
  - Berechnung des Staukontrollfensters (= Überlastungsfenster):
    - Neues Überlastungsfenster += 1 Segment
    - Weiterhin gilt:

      Sendekredit = min {Überlastungsfenster, Empfangsfenster}
  - Es tritt kein Problem auf
    - Überlastfenster steigt bis zum Empfangsfenster und bleibt dann konstant
    - Bei Änderung des Empfangsfensters wird Überlastfenster angepasst

Staukontrolle bzw. Überlastkontrolle

- ACK wird nicht empfangen
  - Man geht davon aus, dass ein weiterer Sender hinzugekommen ist
  - Mit diesem neuen Sender muss man die Pfadkapazität teilen
  - Der Schwellwert wird um die Hälfte der aktuellen Segmentanzahl reduziert
  - Die Segmentanzahl wird wieder auf das Minimum heruntergesetzt

- Die Timerlänge ist entscheidend!
  - Zu lang: Evtl. Leistungsverlust
  - Zu kurz: Erhöhte Last durch erneutes Senden
  - Dynamische Berechnung anhand der Umlaufzeit eines Segments (vgl. Zitterbart)

Fast Recovery nach RFC 2581 (siehe auch implizites NAK)

- Ergänzendes Verfahren zur Staukontrolle: **Fast-Recovery-Algorithmus**
- Empfang von 4 Quittierungen (drei ACK-Duplikate) für eine TCP-PDU veranlasst sofortige Sendewiederholung
- Die Sendeleistung wird entsprechend angepasst
- Der Schwellwert wird auf die Hälfte des aktuellen Staukontrollfensters reduziert
- Da nur von einem Paketverlust und nicht von einem Stau im Netzwerk ausgegangen wird, wird das Staukontrollfenster auf einen Wert über dem Schwellwert eingestellt

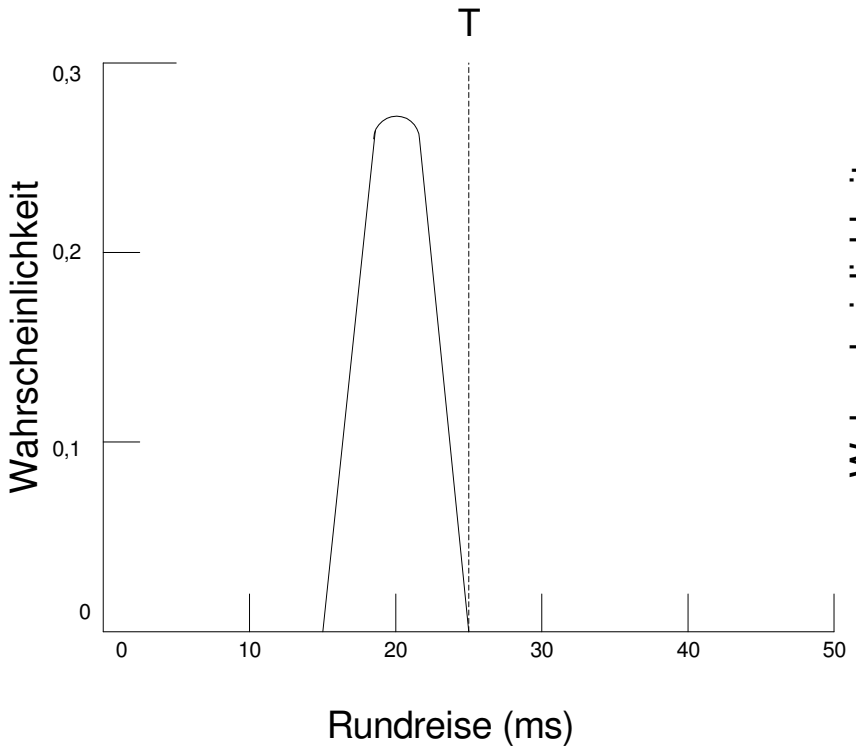
Überblick

1. **TCP (Transmission Control Protocol)**
   - Sliding-Window-Mechanismus
   - Optimierungen: Algorithmen von Nagle und Clark, Silly Window Syndrom
   - Staukontrolle
   - **TCP-Timer**
   - TCP-Zustandsautomat
2. UDP (User Data Protocol)
   - Einordnung und Aufgaben des Protokolls
   - Der UDP-Header
   - Datenübertragung

TCP-Timer

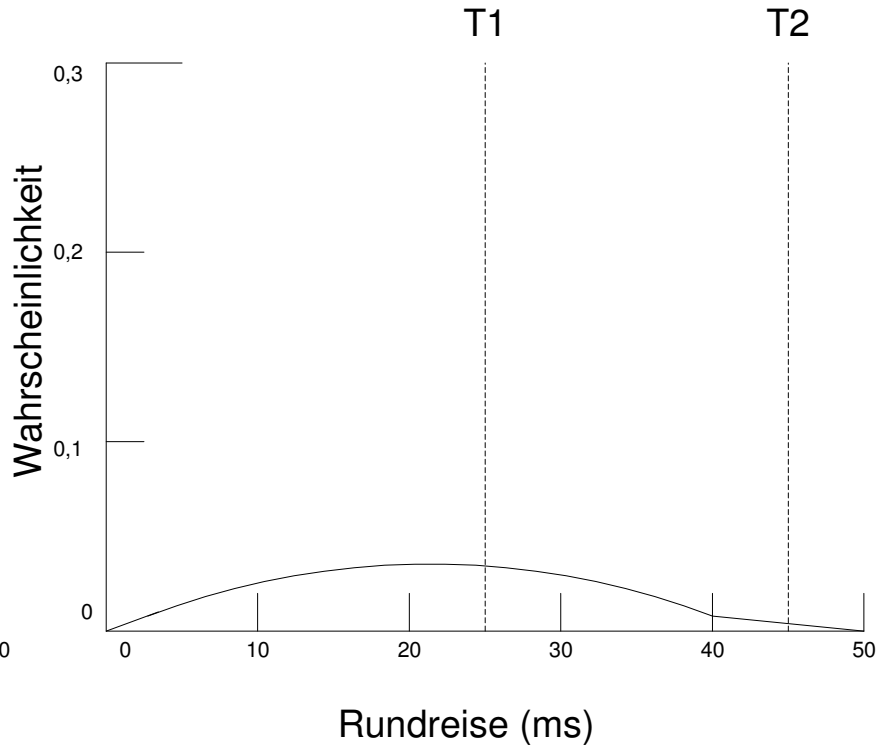
- TCP verwendet einige Timer:
  - **Retransmission Timer**
    - Zur Überwachung der TCP-Segmente nach Karn-Algorithmus
    - Wenn Timer abläuft, wird er verdoppelt und das Segment erneut gesendet (Wiederholung)
  - **Keepalive Timer**
    - Partner wird versucht zu erreichen
    - Gelingt es, bleibt Verbindung bestehen
  - **Timed Wait Timer**
    - Timer für Verbindungsabbau
    - Läuft über doppelte max. Paketlaufzeit (Default: 120 s)
    - Stellt sicher, dass alle gesendeten Pakete nach einem Disconnect-Request noch ankommen

TCP-Timer



(a)

Optimale Länge des Timers T



(b)

Schlechte Timer T1 (zu kurz) und T2 (zu lang)

Überblick

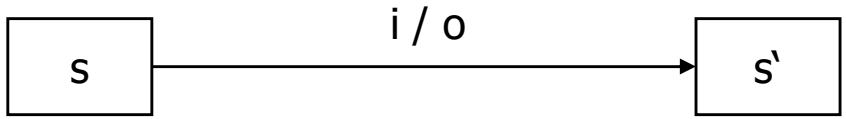
1. **TCP (Transmission Control Protocol)**
   - Sliding-Window-Mechanismus
   - Optimierungen: Algorithmen von Nagle und Clark, Silly Window Syndrom
   - Staukontrolle
   - TCP-Timer
   - **TCP-Zustandsautomat**
2. UDP (User Data Protocol)
   - Einordnung und Aufgaben des Protokolls
   - Der UDP-Header
   - Datenübertragung

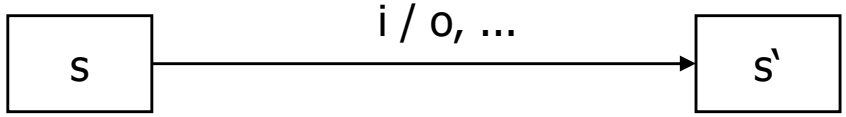
Exkurs: Endliche Zustandsautomaten (1)

- Finite State Machine (FSM)
  - Deterministischer endlicher Automat mit Ausgabe (siehe Mealy-Automat)
  - Verwendet man gerne zur groben Beschreibung des Verhaltens von Protokollinstanzen
  - Ein endlicher Zustandsautomat lässt sich als Quintupel **<S, I, O, T, s_o>** beschreiben:
    - S - endliche, nicht leere Menge von **Zuständen**
    - I - endliche, nicht leere Menge von **Eingaben**
    - O - endliche, nicht leere Menge von **Ausgaben**
    - $T \subseteq S \times (I \cup \{\tau\}) \times O \times S$ – eine **Zustandsüberführungs-funktion**
      - τ bezeichnet eine leere Eingabe
    - $s_o \in S$ – **Initialzustand** des Automaten

Exkurs: Endliche Zustandsautomaten (2)

- Eine Transition (Zustandsübergang) t ∈ T ist definiert durch das Quadrupel **<s, i, o, s‘>** wobei
  - s ∈ S der aktuelle Zustand,
  - i ∈ I eine Eingabe,
  - o ∈ O eine zugehörige Ausgabe und
  - s‘ ∈ S der Folgezustand ist
- Grafische Darstellung eines Zustandsübergangs

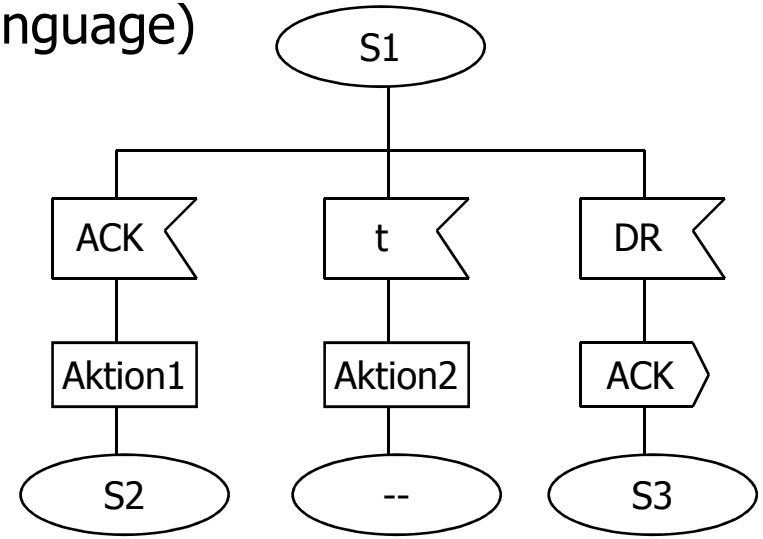




Erweitert: mehrere Ausgaben

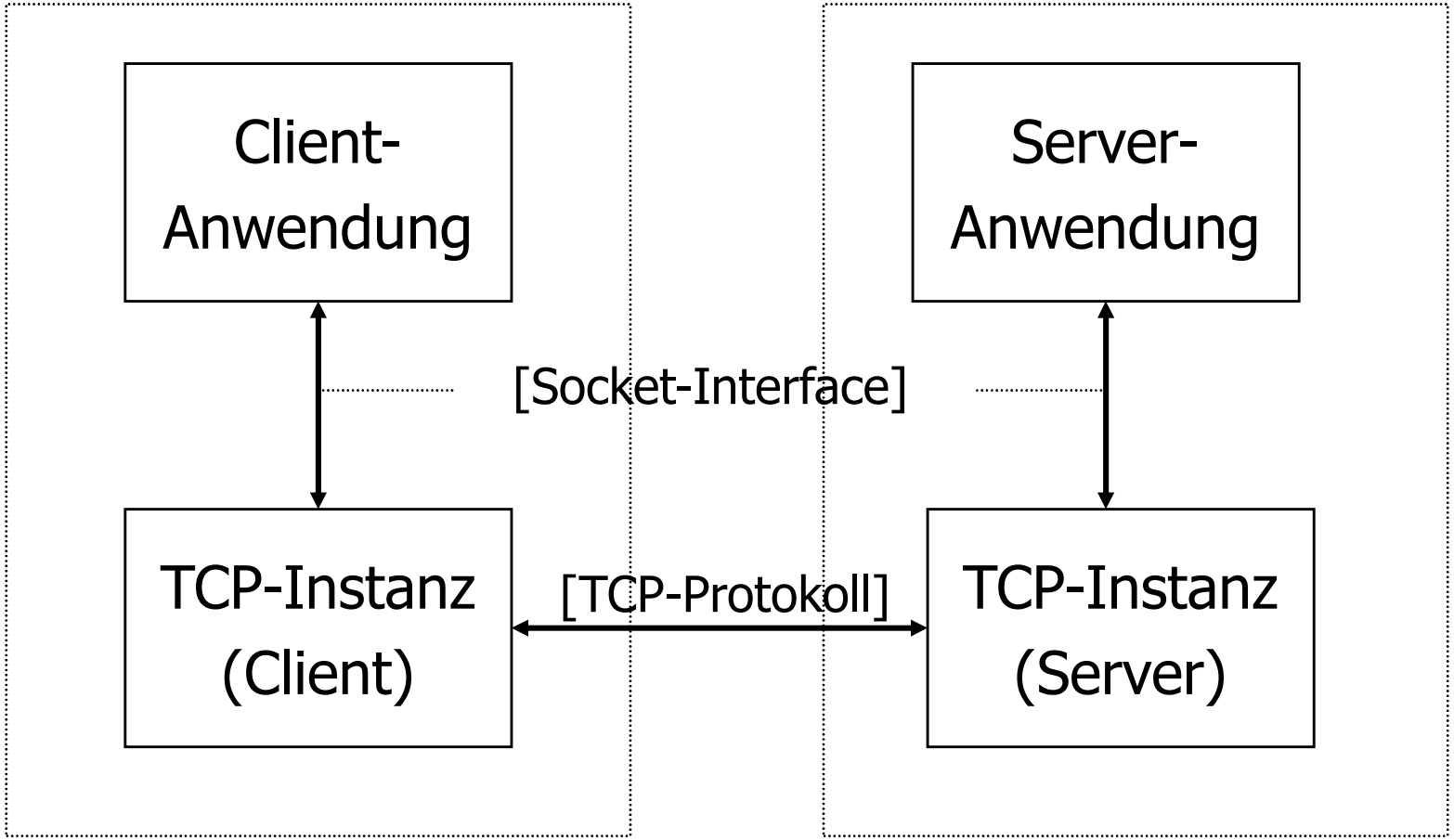
Exkurs: Endliche Zustandsautomaten (3)

- **Nachteil** von FSM: Keine weiteren Zustandsinformationen z.B. in Variable modellierbar
- Daher in der Praxis oft Nutzung **erweiterter endlicher Automaten** (EFSM) für die Detailspezifikation, um Zustandskontexte noch besser zu beschreiben → nutzt weitere Variable neben Zustandsvariable
- Modellierung z.B. in der Sprache **SDL** (Spezification and Description Language)
- Beispiel:



- Kontrollflüsse
- Variablen
- Aktionen
- ..

Zustandsautomat



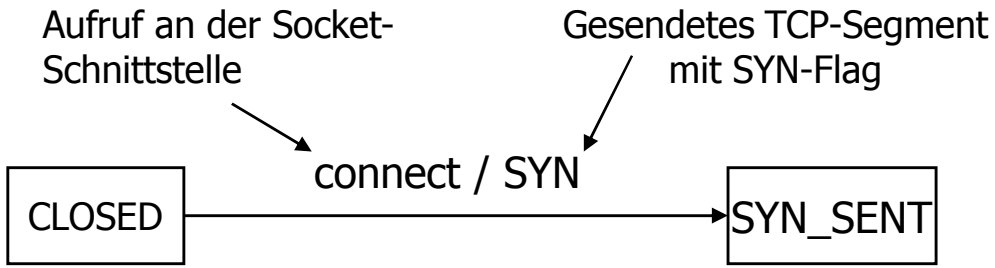
Je ein Zustandsautomat pro Transportverbindung wird
in jeder beteiligten TCP-Instanz verwaltet

Zustandsautomat

Zustand	Beschreibung
CLOSED	Keine Verbindung aktiv oder anstehend
LISTEN	Der Server wartet auf eine ankommende Verbindung
SYN_RCVD	Ankunft einer Verbindungsanfrage und Warten auf Bestätigung
SYN_SENT	Die Anwendung hat begonnen, eine Verbindung zu öffnen
ESTABLISHED	Zustand der normalen Datenübertragung
FIN_WAIT_1	Die Anwendung möchte die Übertragung beenden, Close-Aufruf wurde bereits abgesetzt
FIN_WAIT_2	Die andere Seite ist einverstanden, die Verbindung abzubauen, Bestätigung (ACK) gesendet
TIME_WAIT	Warten, bis keine Segmente mehr kommen
CLOSING	Beide Seiten haben versucht, gleichzeitig zu beenden
CLOSE_WAIT	Die Gegenseite hat den Abbau eingeleitet, warten auf Close-Aufruf der lokalen Anwendung
LAST_ACK	Warten, bis letzte Bestätigung (ACK) für Verbindungsabbau angekommen ist

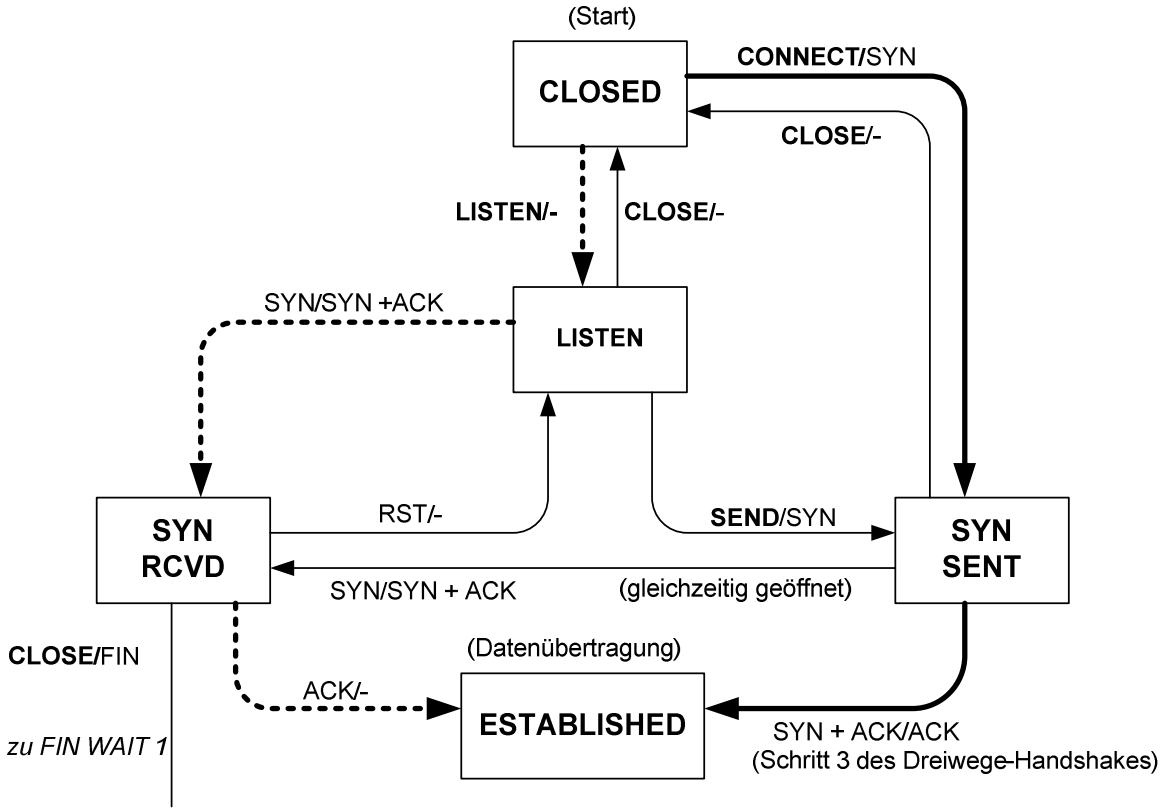
TCP als FSM

- Ein TCP-Zustandsautomat lässt sich als Quintupel **<S, I, O, T, s_o>** beschreiben:
  - S = {CLOSED, LISTEN, SYN_RCVD, ...}
  - I = {connect, send, close, SYN, ACK, FIN, ...}
  - O = {SYN, ACK, FIN, ...}
  - s_o = CLOSED $\in$ S
- Hinweis: Wir beschreiben im Weiteren nur die Transitionen des Verbindungsauf- und abbaus
- Beispiel einer Transition:

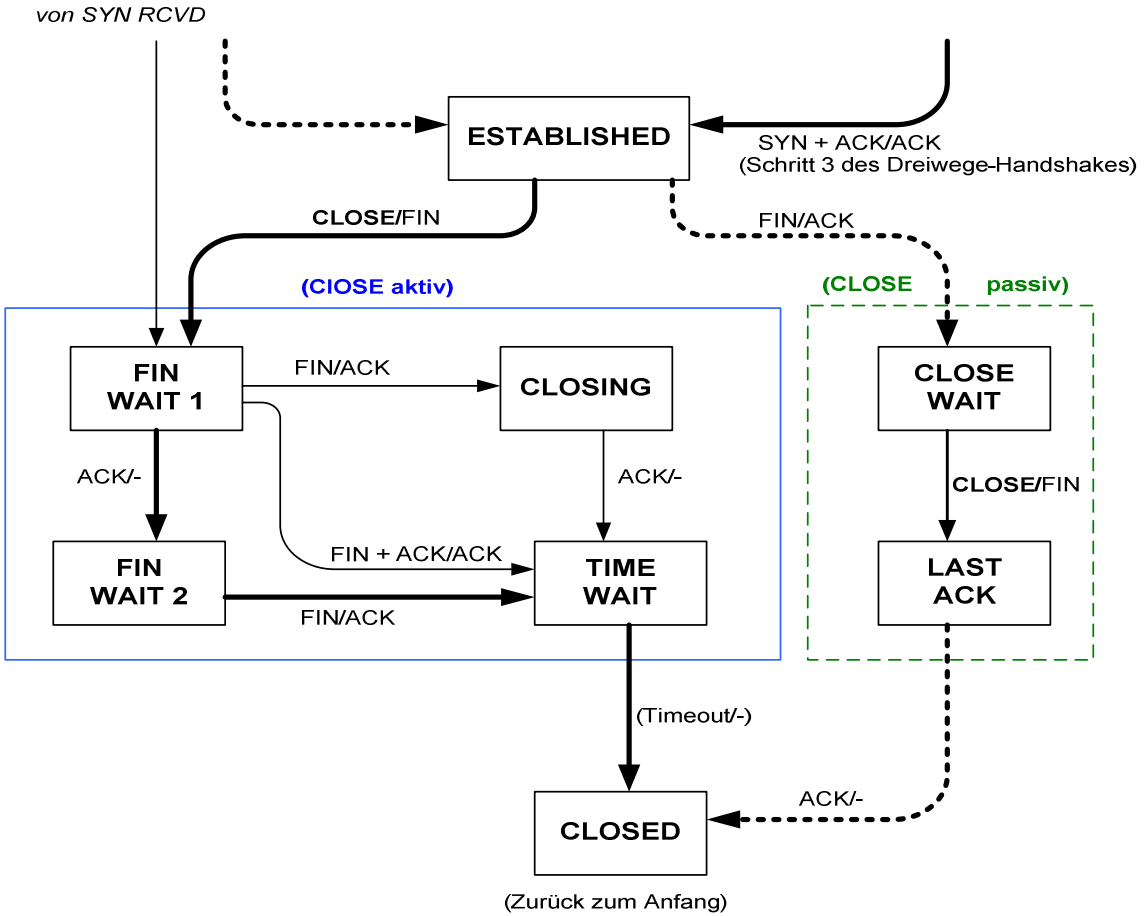


Finite-State-Machine-Modell (1)

fette Linie: normaler Pfad des Clients
fette gestrichelte Linie: normaler Pfad des Servers
feine Linie: ungewöhnliche Ereignisse

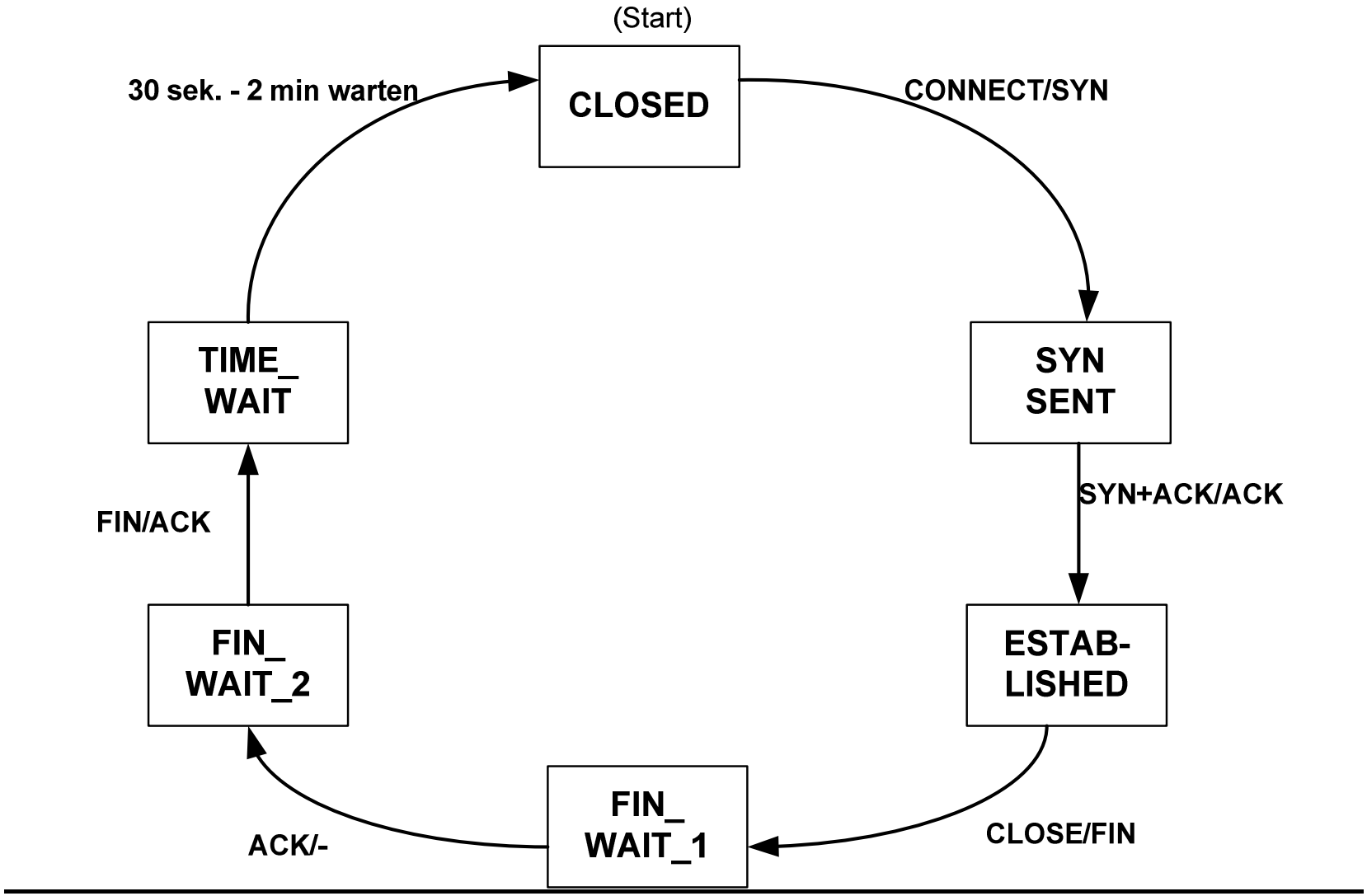


Finite-State-Machine-Modell (2)

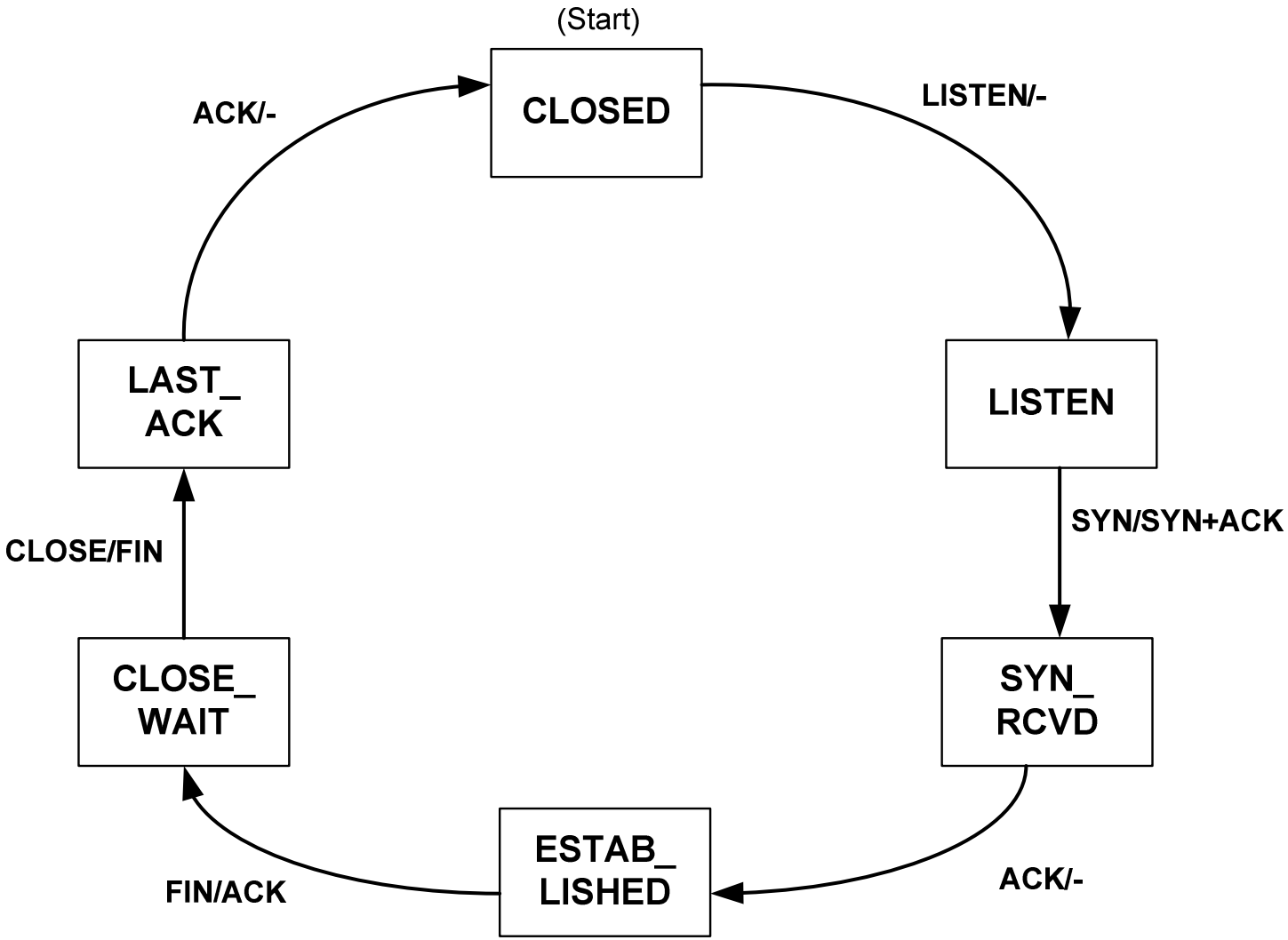


- Es gibt vier Möglichkeiten, die Verbindung abzubauen. Welche?
- 1) Close Aktiv 2) Close passiv 3) Beide gleichzeitig (FIN WAIT1 → CLOSING → TIME WAIT → CLOSED) 4) Selten: Close auf passiver Seite bei FIN schon da

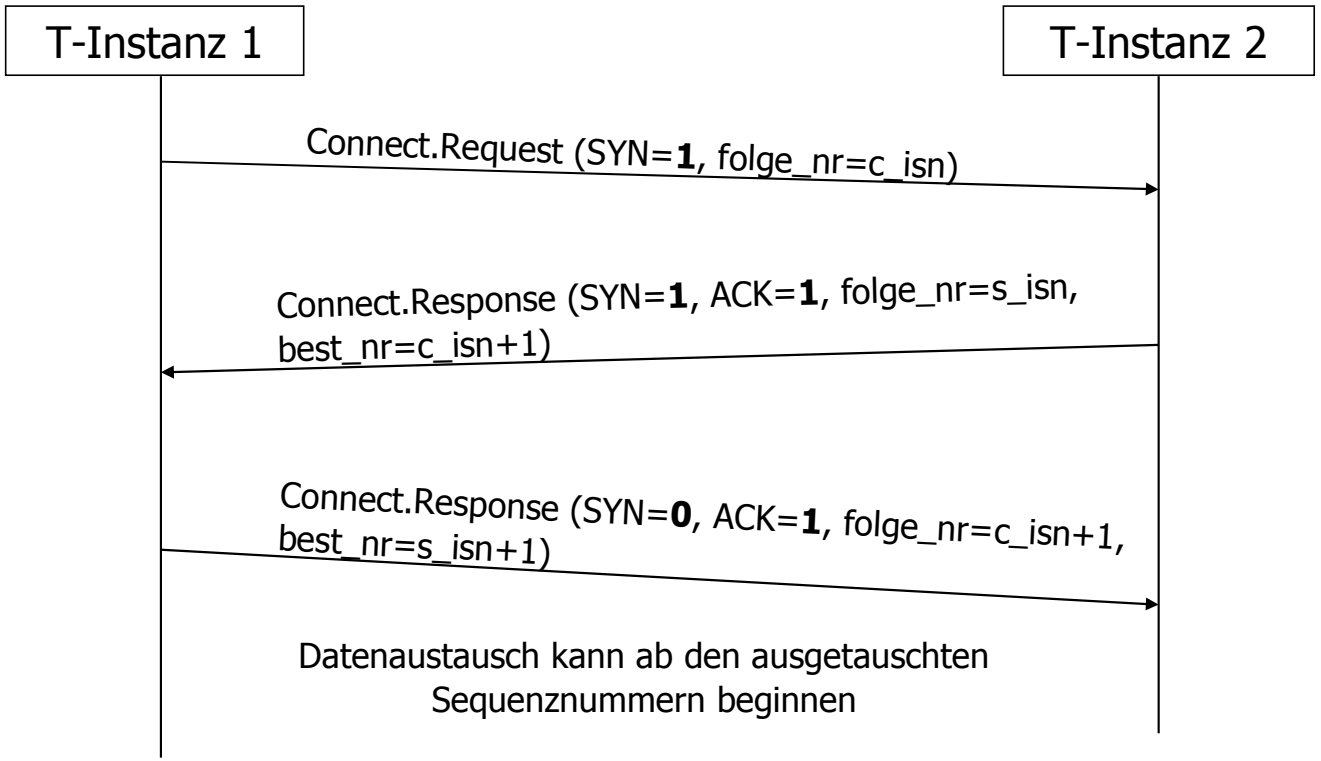
Finite-State-Machine-Modell, TCP-Client (vereinfacht)



Finite-State-Machine-Modell, TCP-Server (vereinfacht)



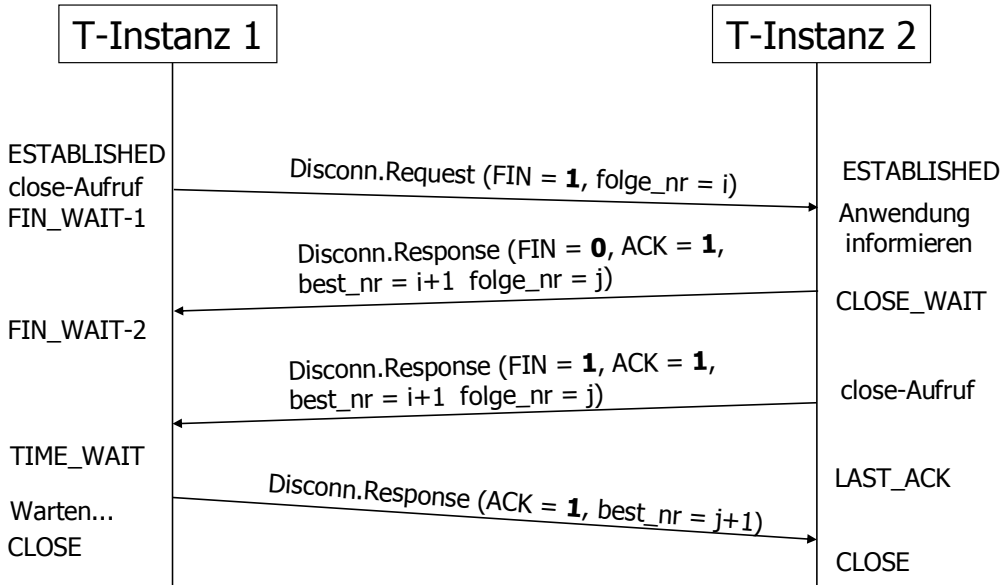
Verbindungsaufbau



c_isn = Initial Sequence Number des Clients (Instanz 1)
s_isn = Initial Sequence Number des Servers (Instanz 2)

Verbindungsabbau

- Client baut die Verbindung ab (auch Server kann es)
- Alle Segmente mit Folgenummer < i bzw. j sind noch zu verarbeiten



Zustände im TCP-Zustandsautomat:
ESTABLISHED, FIN WAIT-1, FIN WAIT-2, TIMED WAIT, CLOSE, CLOSE WAIT, LAST ACK

Übung

- Wozu dienen folgende Stati des TCP-Zustandsautomaten:
  - SYN_RECVD
  - SYN_SENT
  - TIME_WAIT
  - CLOSE_WAIT
- Betrachten Sie mit dem Kommando **netstat** die Zustände diverser TCP-Verbindungen, die auf Ihrem Rechner laufen
  - Web-Browser
  - Chat-Anwendung
  - ...

Überblick

1. TCP (Transmission Control Protocol)

- Sliding-Window-Mechanismus
- Optimierungen: Algorithmen von Nagle und Clark, Silly Window Syndrom
- Staukontrolle
- TCP-Timer
- TCP-Zustandsautomat

2. UDP (User Data Protocol)

- Einordnung und Aufgaben des Protokolls
- Der UDP-Header
- Datenübertragung

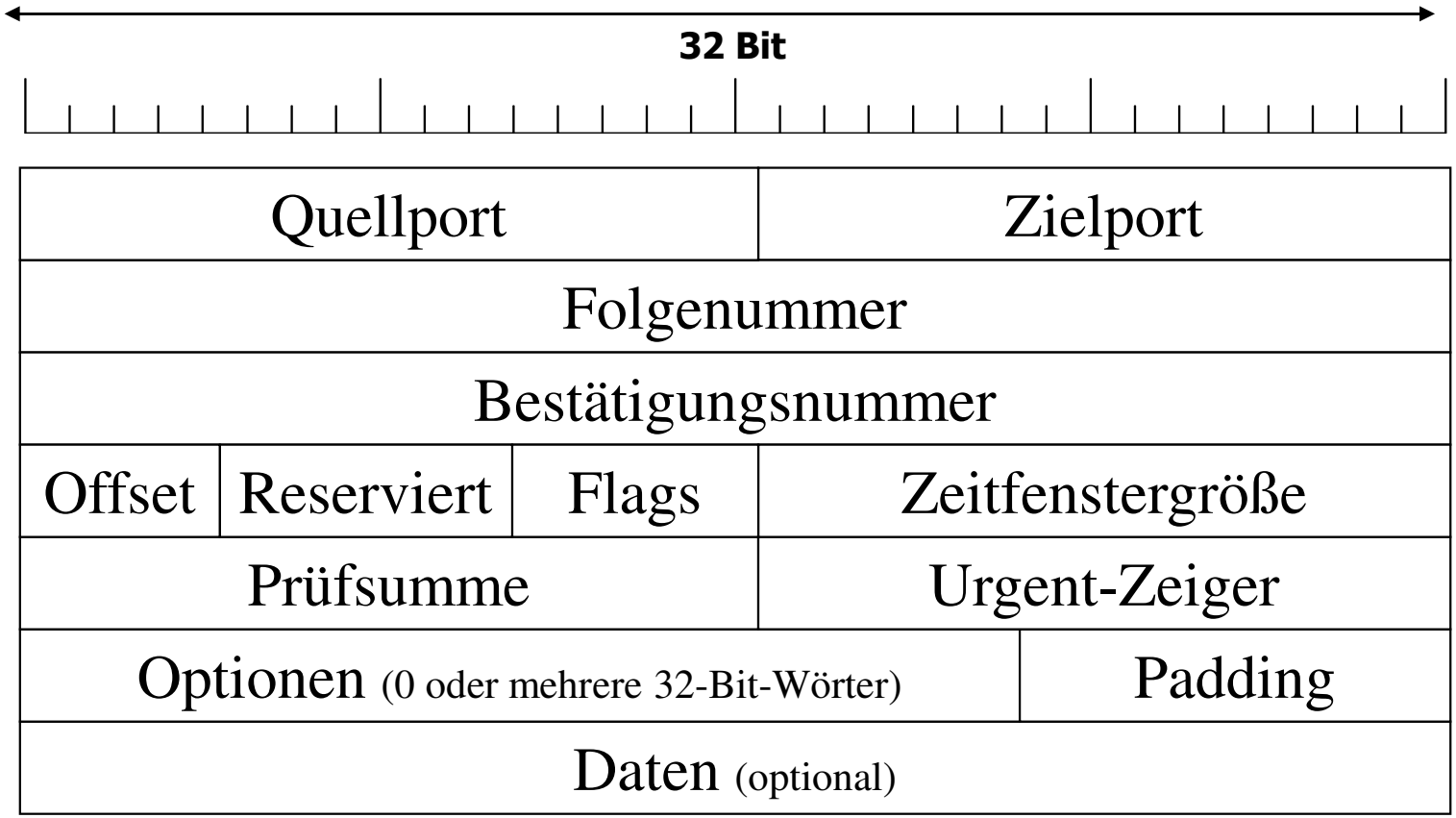
Einordnung und Aufgaben

- **Unzuverlässiges**, verbindungsloses Transport-protokoll
  - **Keine Empfangsbestätigung** für Pakete
  - UDP-Nachrichten **können** ohne Kontrolle **verloren gehen**
  - Eingehende Pakete werden **nicht in einer Reihenfolge sortiert**
  - Maßnahmen zur Erhöhung der Zuverlässigkeit müssen im Anwendungsprotokoll ergriffen werden, z.B.
    - ACK und Warten mit Timeout
    - Wiederholtes Senden bei fehlendem ACK
- **Vorteile** von UDP gegenüber TCP
  - Bessere Leistung möglich, aber nur, wenn TCP nicht nachgebaut werden muss
  - Multicast- und Broadcast wird unterstützt

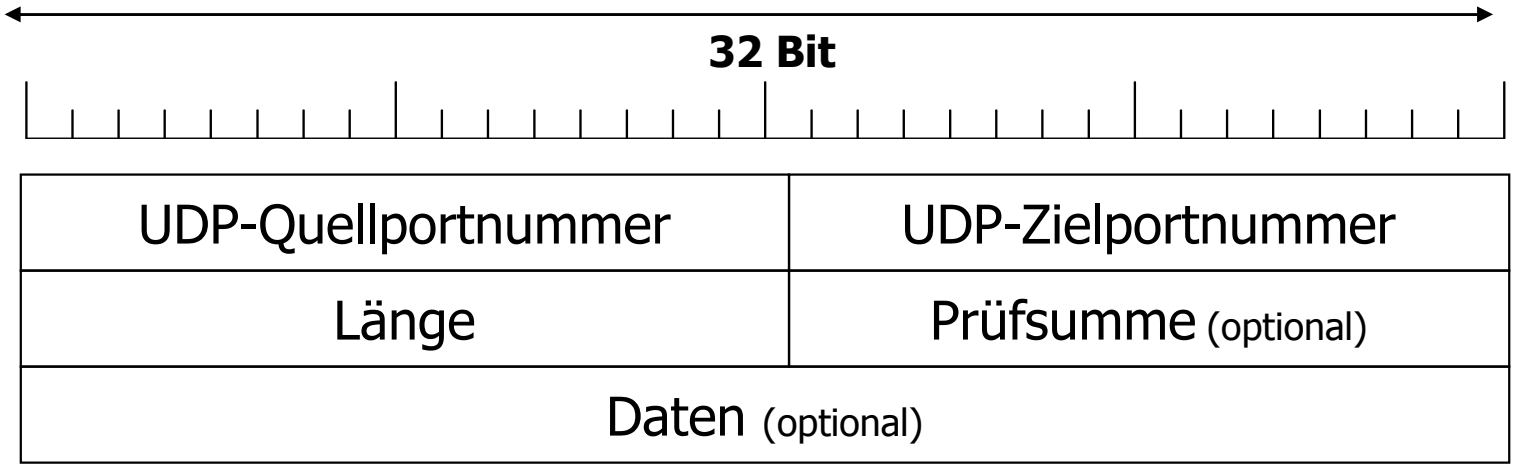
Einordnung und Aufgaben

- Bei UDP ist **keine explizite Verbindungsaufbau-Phase** erforderlich und entsprechend auch **kein Verbindungsabbau**
- Userprozess erzeugt ein UDP-Socket und kann Nachrichten senden und empfangen
- Nachrichten werden bei UDP als **Datagramme** bezeichnet
- In den Datagrammen wird die T-SAP-Adresse des Senders und des Empfängers gesendet (UDP-Ports)

TCP-Header (PCI, Protocol Control Information)



UDP-Header (PCI)

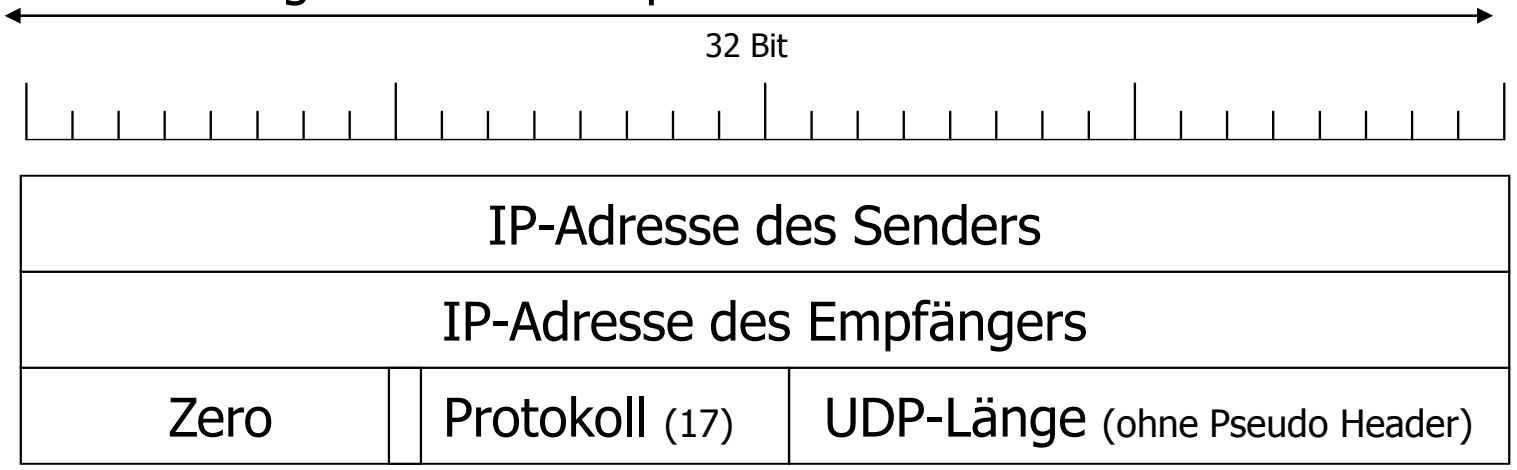


UDP-Header

- **UDP-Ziel-Portnummer**: Nummer des empfangenden Ports
- **UDP-Quell-Portnummer**: Nummer des sendenden Ports
- **Länge**: Größe des UDP-Segments inkl. Header in Byte
- **Prüfsumme (optional):** Prüft das Gesamtsegment (Daten + Header) einschließlich eines Pseudoheaders
- **Daten**: Nettodaten der Nachricht

Pseudoheader (1)

- Die Prüfsumme ist die einzige Möglichkeit, die intakte Übertragung beim Empfänger zu verifizieren
- Vor dem Berechnen der Prüfsumme wird ein Pseudoheader ergänzt
- Das Segment wird auf eine durch 16 Bit teilbare Größe aufgefüllt (gerade Anzahl an Bytes)
- Berechnung auch wie bei TCP über Addition von 16-Bit-Worten und Bildung des Einerkompliments der Summe



Pseudoheader (2)

- Der Empfänger muss bei Empfang einer UDP-Nachricht folgendes unternehmen:
  - die IP-Adressen aus dem ankommenden IP-Paket lesen
  - Der Pseudoheader muss zusammengebaut werden
  - Die Prüfsumme muss ebenfalls berechnet werden
  - Die mit gesendete Prüfsumme mit der berechneten vergleichen
- Wenn die beiden Prüfsummen identisch sind, dann muss das Datagramm seinen Zielrechner und auch den richtigen UDP-Port erreicht haben
  - Ziel des Pseudoheaders ist es somit, beim Empfänger herauszufinden, ob das Paket den richtigen Empfänger gefunden hat

Einige well-known UDP-Portnummern

UDP-Portnummer	Protokoll, Service
53	DNS – Domain Name Service
520	RIP – Routing Information Protocol
161	SNMP – Simple Network Management Protocol
69	TFTP – Trivial File Transfer Protocol

Begrenzte Nachrichtenlänge bei UDP

- Die Länge eines UDP-Segments ist minimal 8 Byte und maximal $2^{16} - 1 = 65.535$ Byte
- Nettodatenlänge = $2^{16} - 1 - 8 = 65.527$ Byte
- Darüber hinaus ist es sinnvoll, die UDP-Segmente nicht länger als in der möglichen IP-Paketlänge zu versenden, da sonst fragmentiert werden muss

Rückblick

1. TCP (Transmission Control Protocol)
   - Sliding-Window-Mechanismus
   - Optimierungen: Algorithmen von Nagle und Clark, Silly Window Syndrom
   - Staukontrolle
   - TCP-Timer
   - TCP-Zustandsautomat
2. UDP (User Data Protocol)
   - Einordnung und Aufgaben des Protokolls
   - Der UDP-Header
   - Datenübertragung