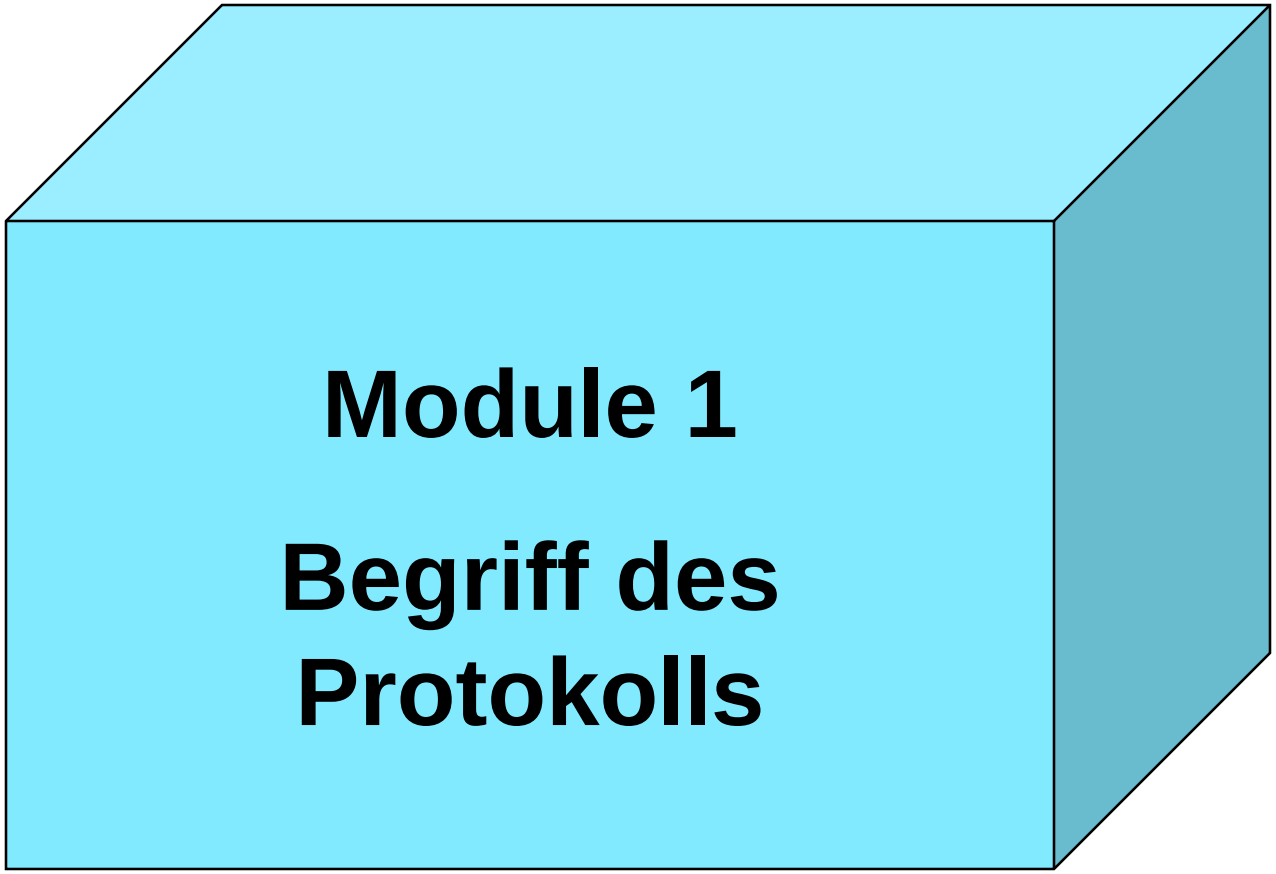
Internetkommunikation – SS 2017

Umfang:	**2 SWS Vorlesung**
	2 SWS Übungen/Praktikum
Start:	**Montag, 10. April 2017**
Teilnehmer:	**BCS, 4. Semester, SPEZ (TK) - Credits: 6.0 von 12.0**
Vorkenntnisse:	**Netze**
Lehrziele:	**Einführung in das Protokolldesign am Beispiel TCP**
Lehrmethodik:	**Vorlesung**
	Übungsblätter und Laborpraktikum zu Inhalten der Vorlesung

Prüfungsvorleistung:

1) **regelmäßige Teilnahme an der Lehrveranstaltung**
2) **regelmäßige schriftliche Bearbeitung der Übungsblätter (→ LEA)**
3) **Vorrechnen und Diskussion von Übungsaufgaben**
4) **Aktive Teilnahme am Laborpraktikum**



Begriff des Protokolls

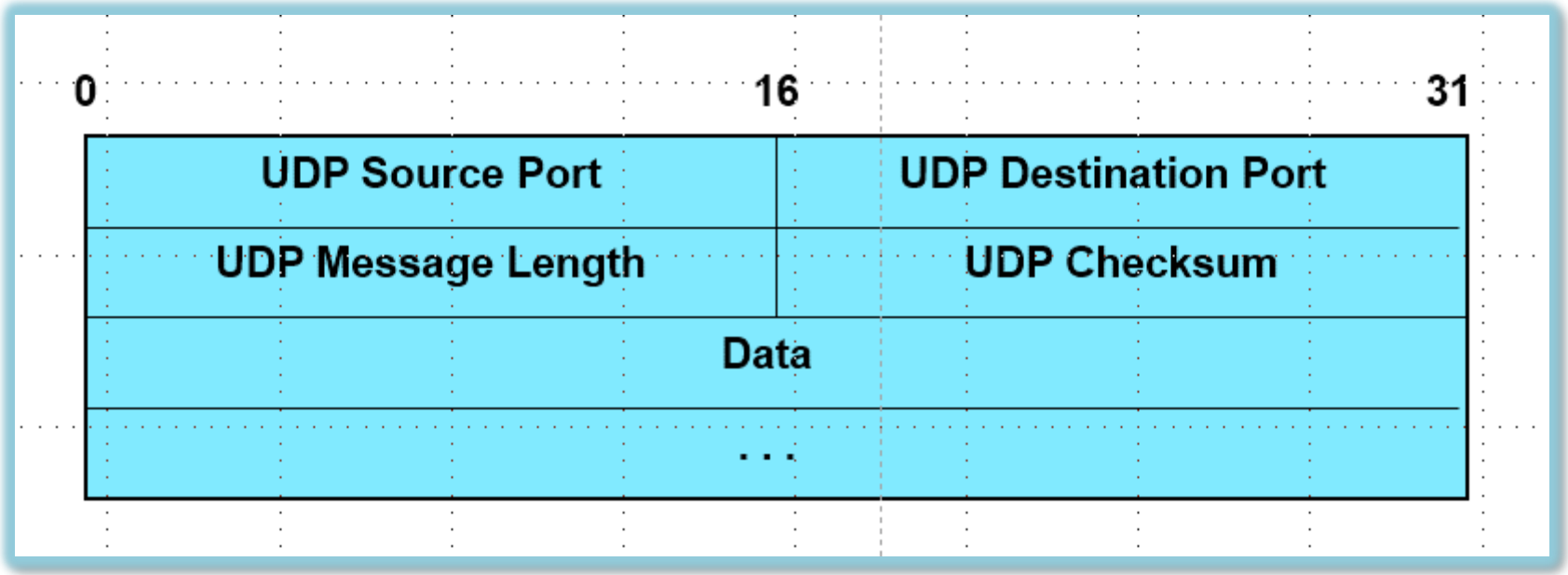
Ein Protokoll definiert sich über

- **Syntax**
  Frage: Wie sind die PDUs syntaktisch aufgebaut?
  Werkzeug: Byteweise Definition der Datenstrukturen, formale Sprache (ASN.1, XML)
- **Semantik**
  Frage: Was bedeuten die PDUs?
  Nach welchen Regeln läuft das Protokoll ab?
  Werkzeug: oft umgangssprachliche Formulierung,
  bisweilen auch semantische Modelle
- **Timing**
  Frage: Wann passiert was? Wie sind die zeitlichen Zusammenhänge?
  Werkzeug: Timer-Vorgaben, steuernde Algorithmen, mathematische Modelle

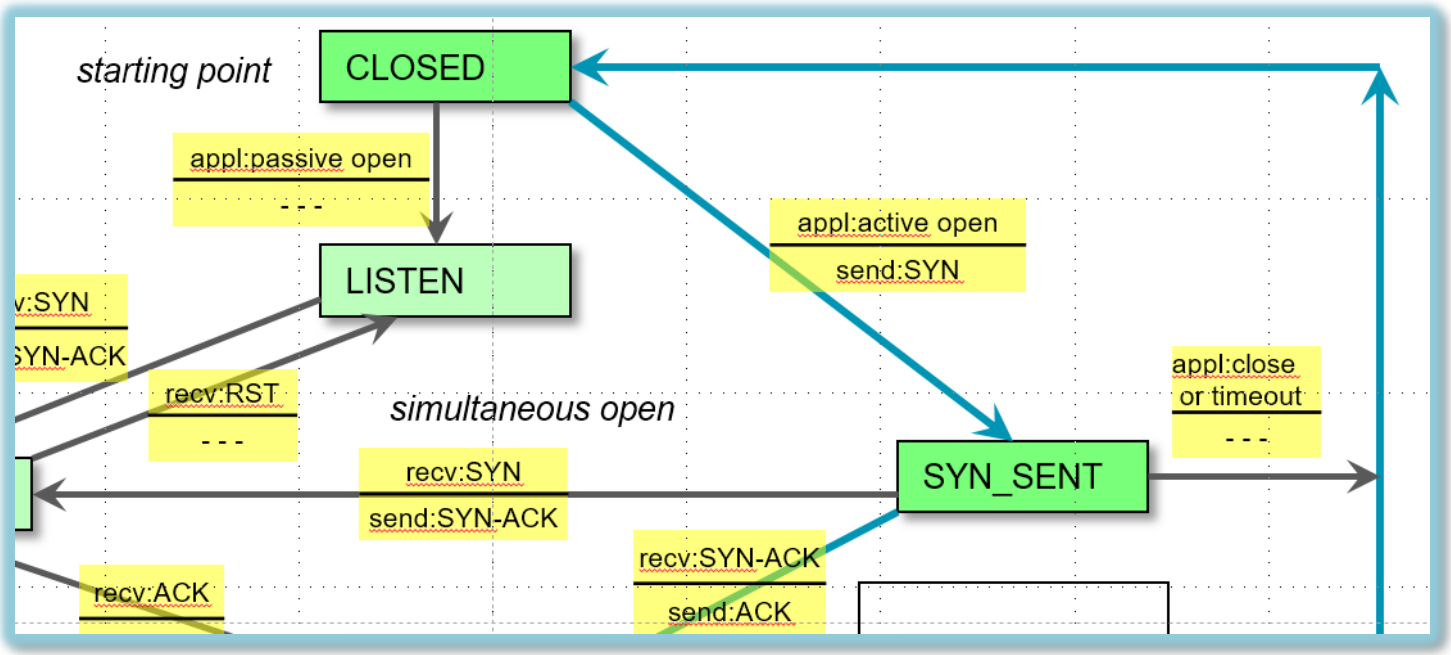
Sehr nützlich in diesem Zusammenhang sind endliche Automaten

- Eingabezeichen → eingehende PDUs, Funktionsaufrufe
- Ausgabezeichen → ausgehende PDUs, Funktionsaufrufe
- Zustandsübergangsfunktion → Regelwerk

Syntax-Darstellung: z.B. durch ein PDU-Block-Diagramm



Semantikdarstellung: z.B. durch einen endlichen Automaten



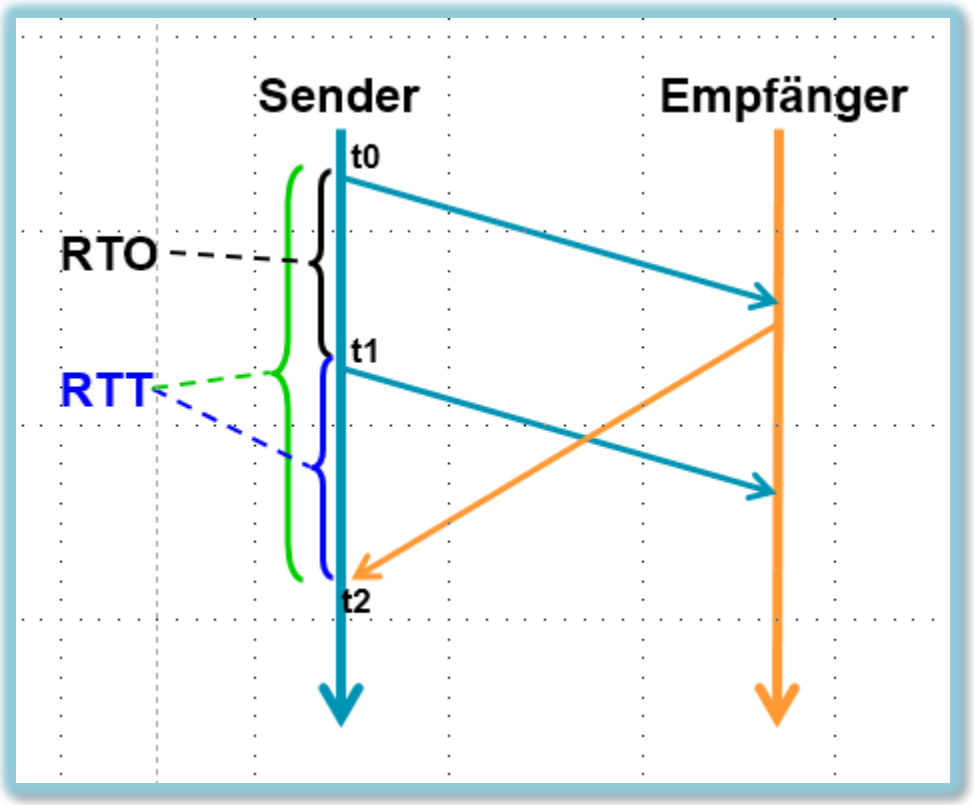
CLOSED — Zustand

⟶ Zustands-übergang

Eingabezeichen

Ausgabezeichen

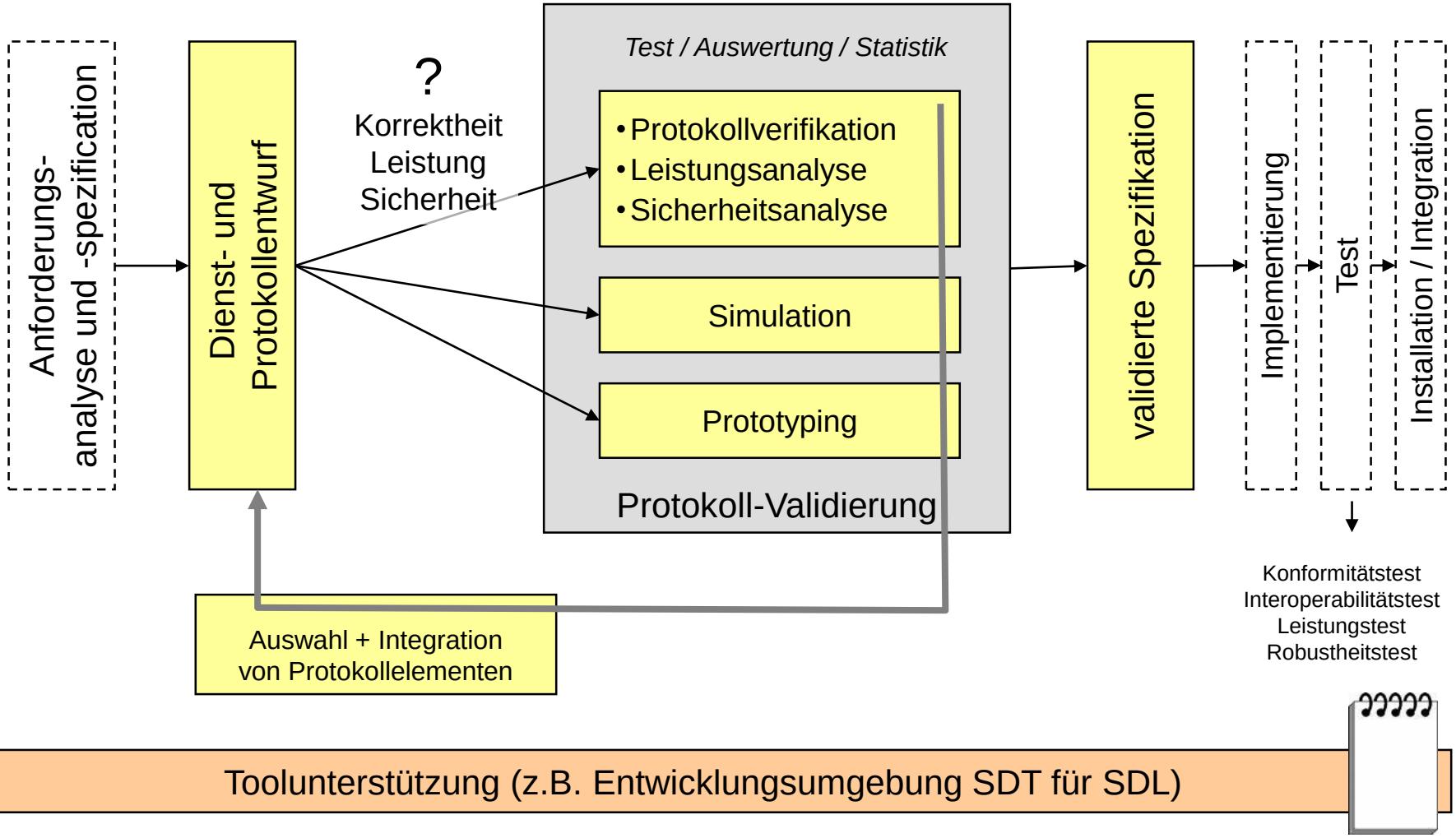
Timing-Darstellung: z.B. durch Protokoll-Zeit-Sequenz-Diagramm



Grundfragen zu Protokollen

- Aufgabe des Protokolls
- Einordnung des Protokolls in das Schichtenmodell
- Spezifikation eines Protokolls
- Grundlegende Eigenschaften des Protokolls (theoretisch / in der Praxis)
- Entwicklung eines Protokolls
- Implementierung des Protokolls
- Testen des Protokolls
- Erweiterungen/Varianten eines Protokolls
- Zusammenwirken mit anderen Protokollen
- Aufbau einer Protokollarchitektur

Entwicklungsprozess für Protokolle



Im folgenden: Beispiel des Stop-and-Wait Protokolls

Protokollverifikation

- **Korrektheitsbetrachtungen zum Stop-and-Wait Protokoll**
- *1. Reflexion: Wie kann man sich überhaupt jemals sicher sein?*
- **Methoden der Protokollverifikation**

Leistungsanalyse

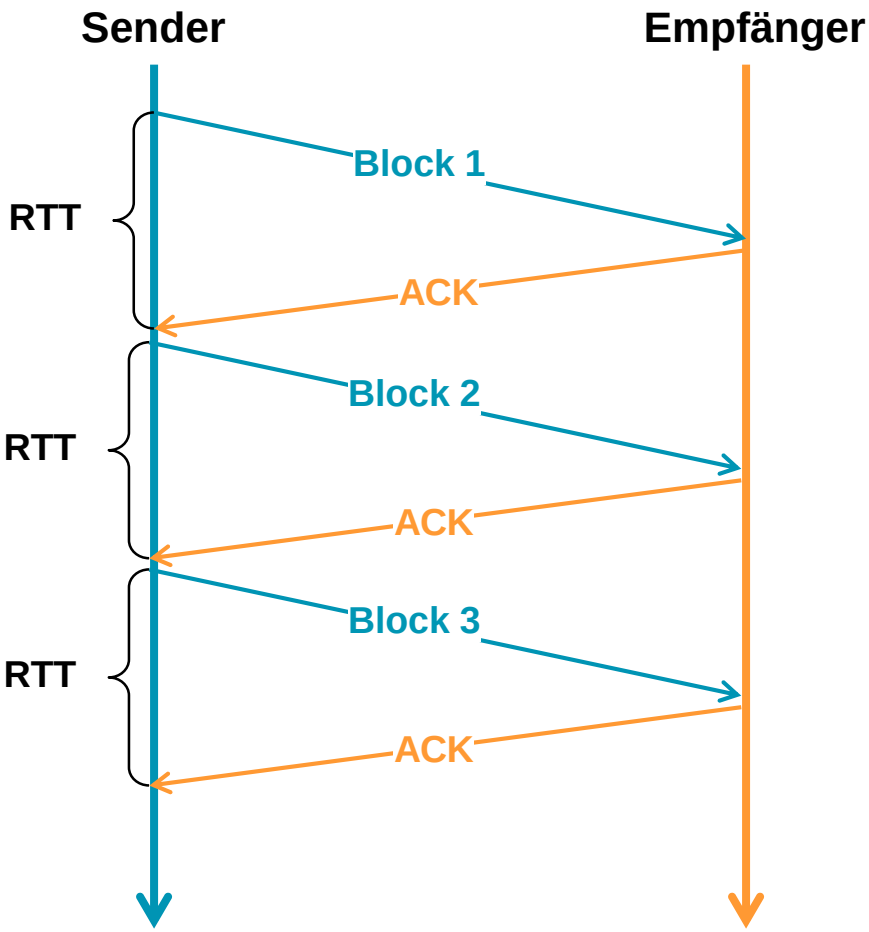
- **Effizienzbetrachtung zum Stop-and-Wait Protokoll (Idealbedingungen)**
- **Effizienzbetrachtung zum Sliding-Window Protokoll (Idealbedingungen)**
- *2. Reflexion: Theorie versus Praxis*
- **Statistische Analyse zum Stop-and-Wait Protokoll**
- *3. Reflexion: Theorie versus Praxis*
- **Experimentelle Analyse zum Stop-and-Wait Protokoll (nur theoretischer Teil)**

Stop-and-Wait Protokoll

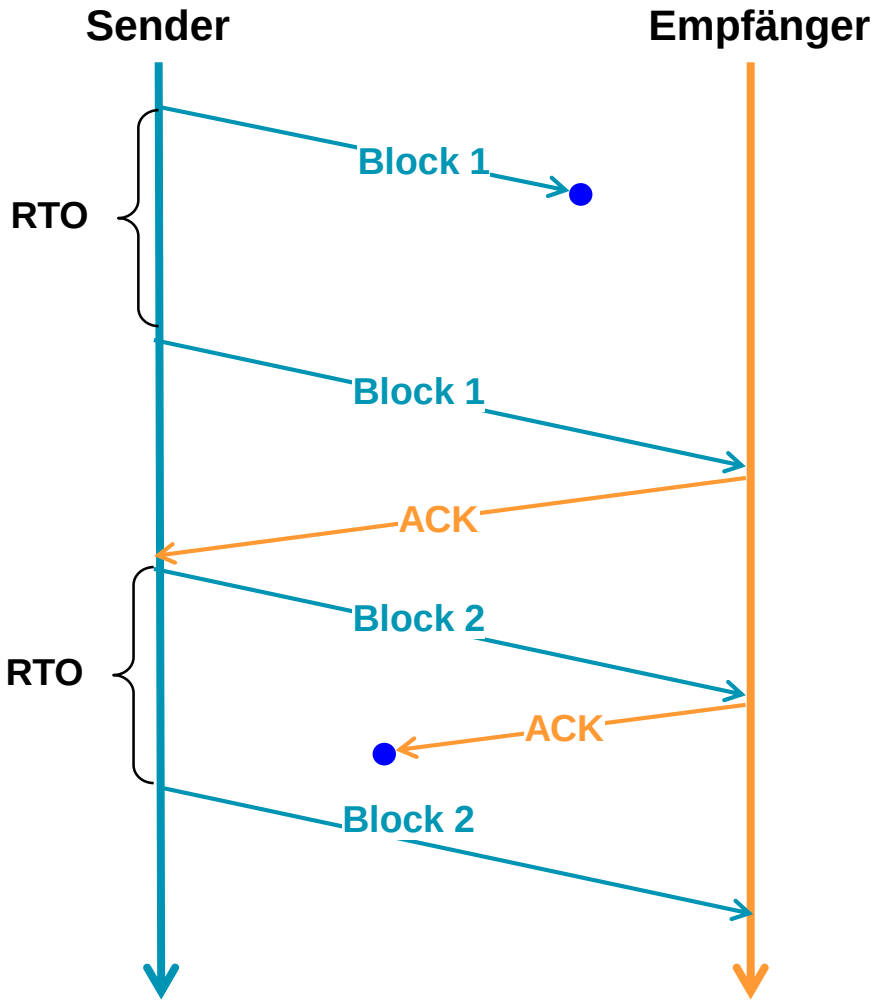
Das einfachstes Protokoll

- zur Sicherung der Daten
- zur Flusssteuerung
- zum Erhalt der Datenreihenfolge bei nichtdeterministischen Übertragungszeiten

Das Stop-and-Wait Protokoll gehört als PAR-Protokoll (Positive Acknowledge and Retransmit) zur Gruppe der ARQ-Protokolle (= Automatic Repeat-reQuest, Automatische Wiederholungsanfrage)



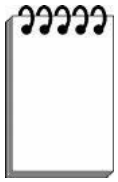
Stop-and-Wait Protokoll



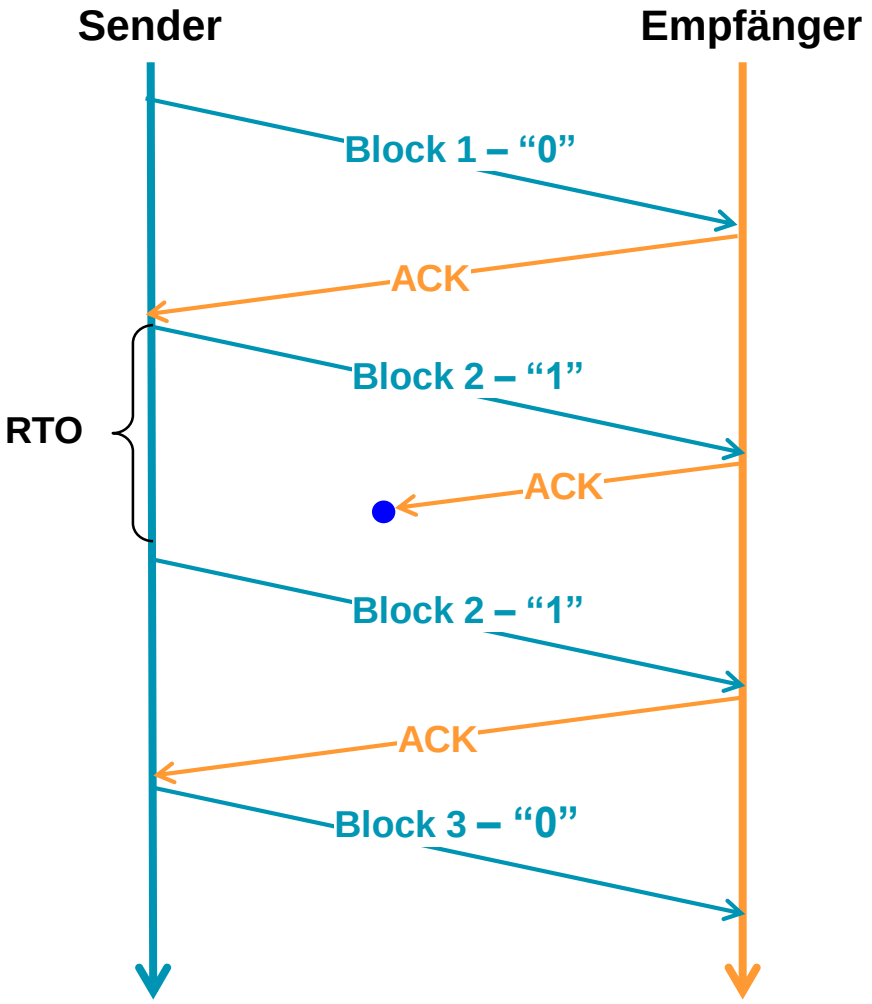
Frage: Was wird mit zweiten Block zwei gemacht ?

Frage: Was bringt NAK ?

Frage: Kann ein zu langsames ACK zu Problemen führen?



Stop-and-Wait Protokoll mit Alternating-Bit

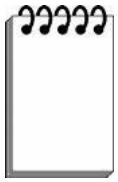


Frage: Genügt ein Bit ?

Frage: Muss mit dem ACK auch ein alternating Bit gesendet werden ?

Frage: Ist das Protokoll jetzt schon korrekt oder sind verwickeltere Abläufe denkbar, die zu Protokollfehlern führen

1. Reflexion



Protokollverifikation – kompakt

- **Was soll durch die Protokollverifikation erreicht werden?**

  Formaler Nachweis, dass spezifizierter Kommunikationsdienst korrekt, vollständig und konsistent erbracht wird.

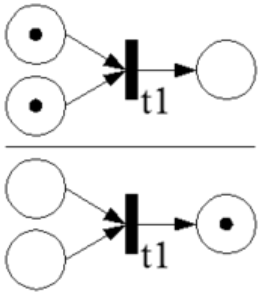
- **Warum ist Protokollverifikation wichtig?**
- **Welche Protokolleigenschaften sollen verifiziert werden?**

  Deadlock-Freiheit, Livelock-Freiheit, Korrektheit, Vollständigkeit, Terminierung, Sicherheit, keine Invariantenverletzung
  weiterhin: Fehlertoleranz, Fairness

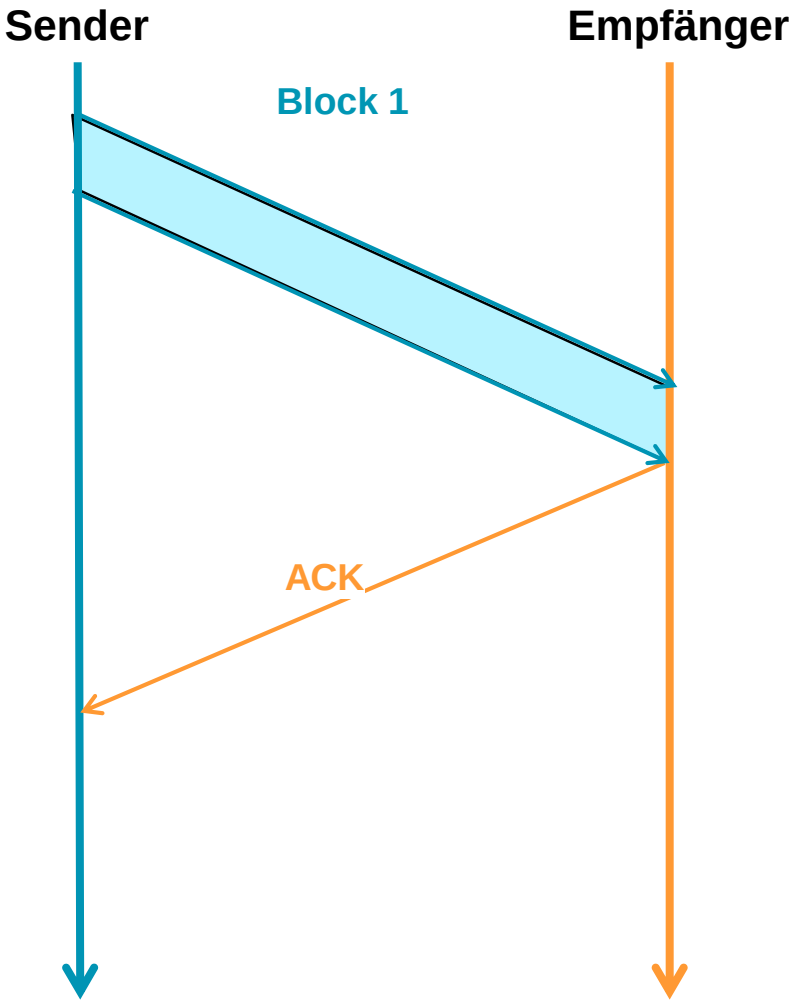
- **Welche grundsätzlichen Formen der Protokollverifikation gibt es?**

  Manuell-mathematische Techniken, logikbasierte Verifikation, modellbasierte Verifikation, hybride Vorgehensweise.

  Wichtige modellbasierte Methoden:
  - Petrinetze
  - Erreichbarkeitsanalyse basierend auf endlichen Automaten (FSM, finite state machine)



Effizienz von Stop-and-Wait sowie Sliding-Window-Protokoll



Parameter:

- Übertragungsrate des Ü-Kanals (Kapazität K)
- Blockgröße B
- Signallaufzeit S vom Sender zum Empfänger (oder umgekehrt)
- Übertragungszeit Ü eines Pakets
- Durchsatz D

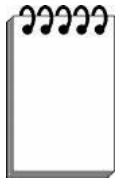
Effizienz := Durchsatz / Kapazität

Sei

a := Signallaufzeit / Übertragungszeit.

(a) Stellen Sie für das Stop-and-Wait-Protokoll die Effizienz E als Funktion von a dar.

(b) Stellen Sie für das Sliding-Window-Protokoll die Effizienz als Funktion von a dar.



Effizienz des Stop-and-Wait-Protokolls

Diskussion der vorhergehenden Folie:

- **Was genau ist Durchsatz?**
  Wie lässt sich der Begriff Durchsatz weiter differenzieren?
- **Von welchen Bedingungen wurde auf der vorhergehenden Folie abstrahiert?**
  Welche Annahmen wurden getroffen?
- **Warum wird überhaupt von der Realität abstrahiert?**
- **Wie weit soll abstrahiert werden bzw. von was soll abstrahiert werden?**
- **Woher weiß ich, dass meine Messergebnisse realitätsrelevant sind?**

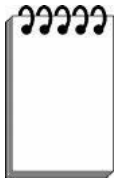
Weiterführende Aufgabenstellungen:

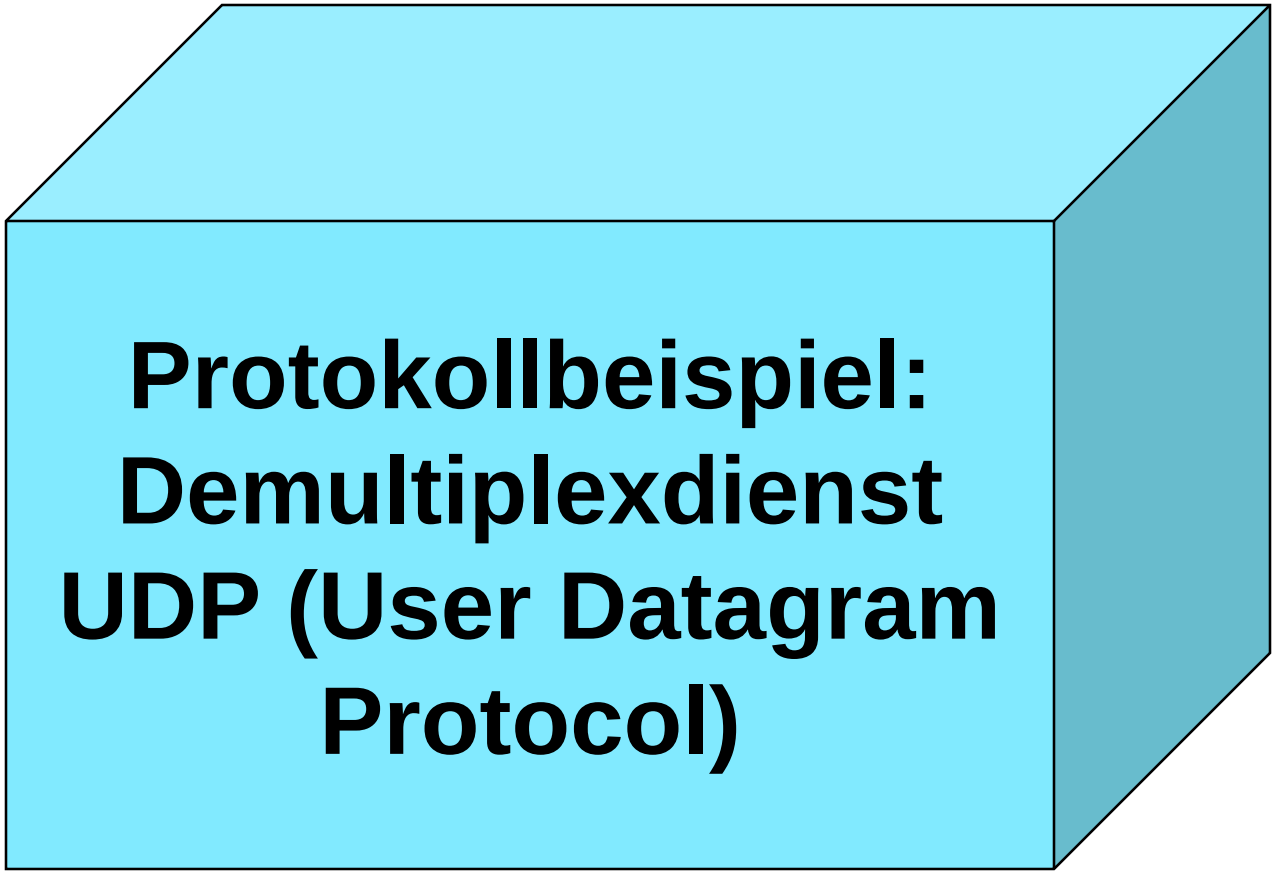
- **Berechnen Sie den Durchsatz des Stop-and-Wait-Protokolls, wenn bei der Übertragung Fehler auftreten können, die eine Retransmission nötig machen.**
- **Stellen Sie hierzu zuerst eine Liste idealisierender Rahmenbedingungen zusammen.**

Effizienz von Stop-and-Wait sowie Sliding-Window-Protokoll

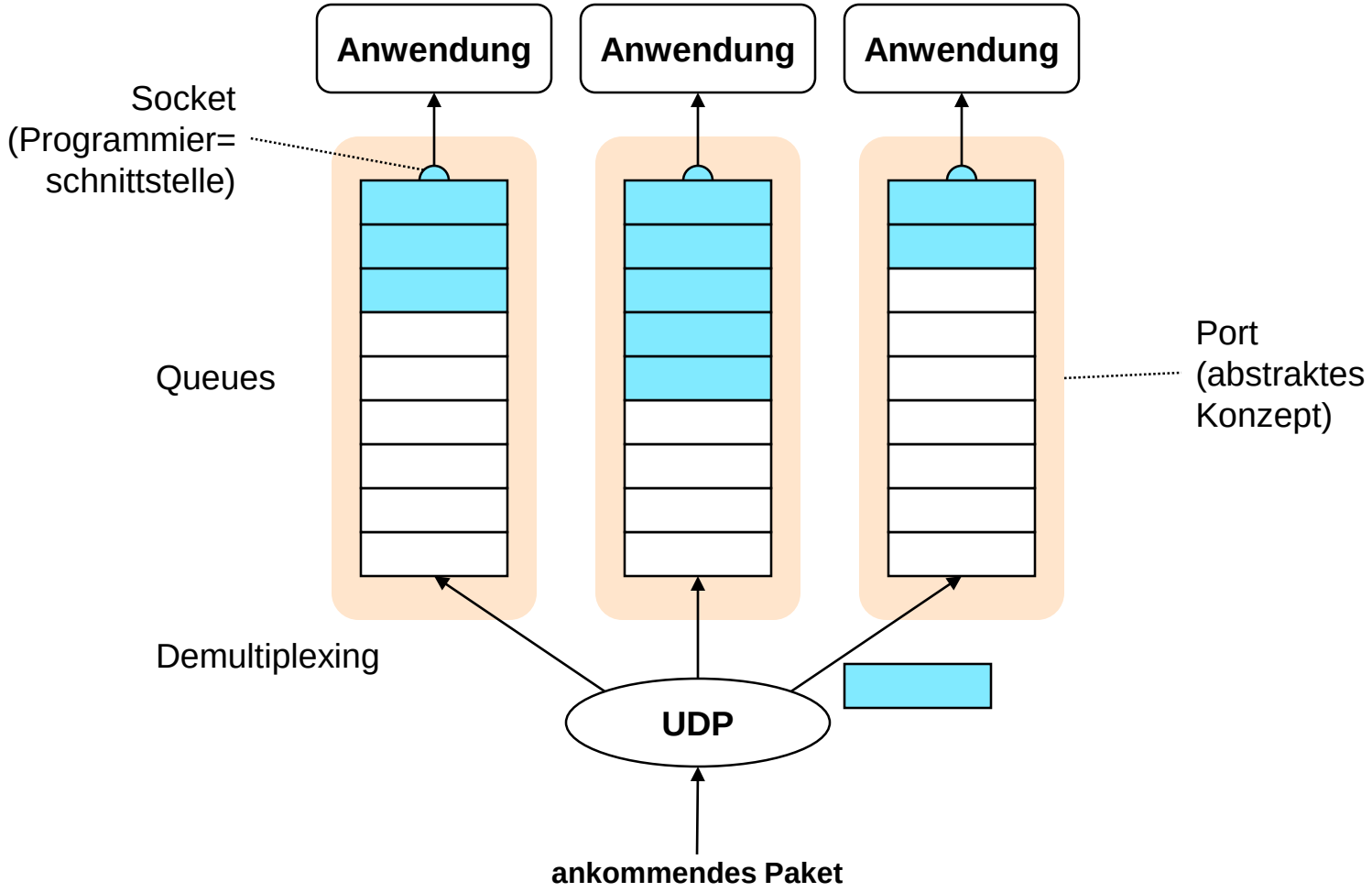
Mögliche Annahmen bei der Berechnung:

- Die Wahrscheinlichkeit, dass ein Bit fehlerhaft ist beträgt p .
- Einzelbitfehler in der Übertragung sind voneinander unabhängig.
- Die Bitfehlerwahrscheinlichkeit p ist unabhängig von der Zeit.
- Der Kanal hat keine Verzögerungszeit.
- Jeder Bitfehler wird sicher erkannt.
- Das ACK wird immer korrekt empfangen.
- der Retransmissiontimer ist immer korrekt und optimal eingestellt.

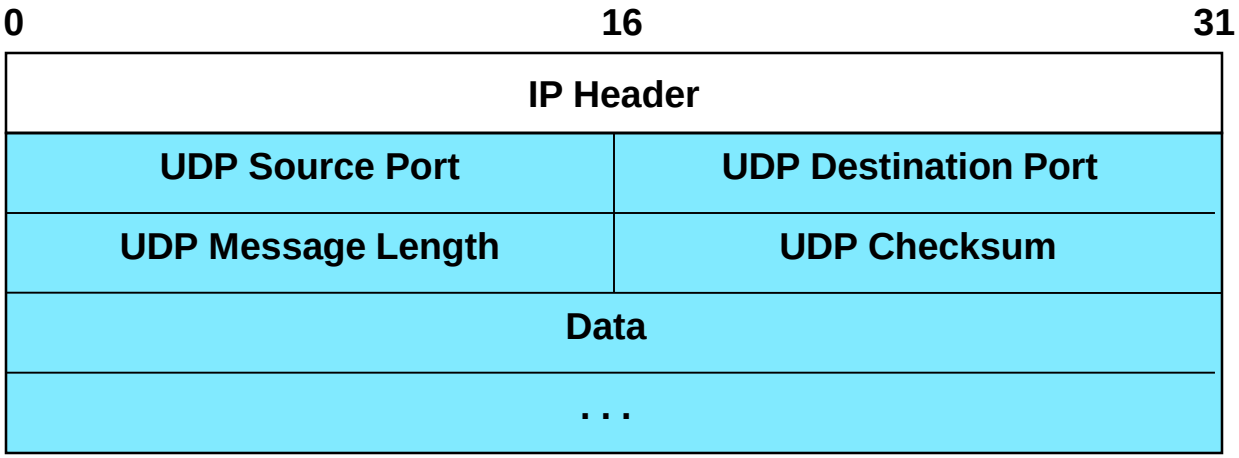



Protokollbeispiel:
Demultiplexdienst
UDP (User Datagram
Protocol)

UDP Demultiplexing (Ports und Sockets)



UDP Header



Source Port:	**Enthält den Port des versendenden Hosts; optionale Angabe, d.h. wenn angegeben, dann ist dies der Port, an den evtl. Antworten gehen**
Destination Port:	**Zielport, wohin das Datagramm geschickt wird**
Message Length:	**Gibt die Gesamtlänge des Datagramms an**
Checksum:	**Prüfsumme über UDP-Header sowie IP-Pseudoheader (Protokollnummer+IP-Adressen) Stellt sicher, dass der richtige Port und der richtige Host erreicht werden.**

Ablauf UDP-Socketkommunikation