Karlsruher Institut für Technologie

Vorlesung Einführung in Rechnernetze
4. Protokollmechanismen

Prof. Dr. Martina Zitterbart

Dipl.-Inform. Martin Florian, Markus Jung (M.Sc.), Matthias Flittner (M.Sc.)
[zitterbart | florian | m.jung | flittner]@kit.edu

Institut für Telematik, Prof. Zitterbart

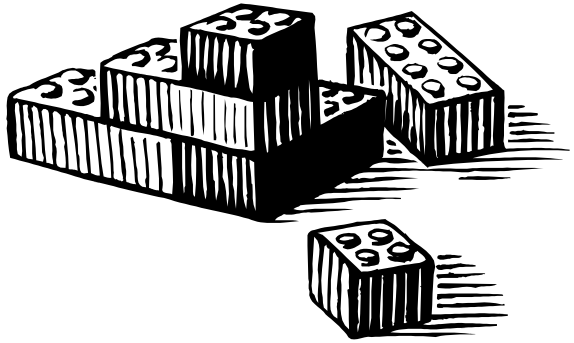
Kapitelübersicht

1. Einführung
2. Netzwerkarchitekturen
3. Physikalische Grundlagen
4. Protokollmechanismen
5. Die Sicherungsschicht: HDLC
6. Die Sicherungsschicht: Lokale Netze
7. Netzkopplung und Vermittlung
8. Die Transportschicht
9. Sicherheit
10. Anwendungssysteme

1. Basis-Szenario
2. Fehlertypen und Fehlerursachen
3. Mechanismen zur Fehler-erkennung und -behebung
4. Fehlerkontrolle bei Bitfehlern
5. Fehlerkontrolle bei Paketfehlern
6. Flusskontrolle
7. Verbindungen
8. Zusammenfassung

… Austausch von Daten

- Ziel
  - Geräte bzw. Anwendungen möchten Daten austauschen
- Protokolle erforderlich, die Formate und Regeln des Datenaustauschs festlegen
  - Protokollmechanismen stellen die Bausteine der Protokolle dar

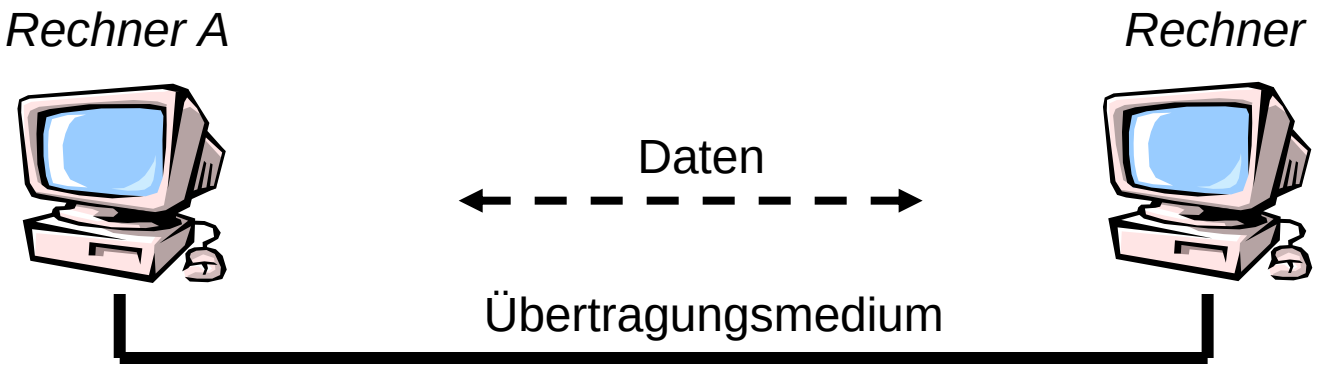


Kapitelübersicht

1. Einführung
2. Netzwerkarchitekturen
3. Physikalische Grundlagen
4. Protokollmechanismen
5. Die Sicherungsschicht: HDLC
6. Die Sicherungsschicht: Lokale Netze
7. Netzkopplung und Vermittlung
8. Die Transportschicht
9. Sicherheit
10. Anwendungssysteme

1. Basis-Szenario
2. Fehlertypen und Fehlerursachen
3. Mechanismen zur Fehler-erkennung und -behebung
4. Fehlerkontrolle bei Bitfehlern
5. Fehlerkontrolle bei Paketfehlern
6. Flusskontrolle
7. Verbindungen
8. Zusammenfassung

4.1 Basis-Szenario



- Ziel
  - Rechner A und Rechner B sind über ein Übertragungsmedium direkt miteinander verbunden und wollen über dieses Daten austauschen
- Probleme
  - Kodierung der Signale auf dem Übertragungsmedium
  - Formate und Regeln für den Datenaustausch erforderlich, d.h. Protokolle
    - Protokollmechanismen stellen die Bausteine der Protokolle dar
  - Bei der Übertragung können Fehler entstehen
    - Ursachen? Erkennung? Behebung?

Kapitelübersicht

1. Einführung
2. Netzwerkarchitekturen
3. Physikalische Grundlagen
4. Protokollmechanismen
5. Die Sicherungsschicht: HDLC
6. Die Sicherungsschicht: Lokale Netze
7. Netzkopplung und Vermittlung
8. Die Transportschicht
9. Sicherheit
10. Anwendungssysteme

1. Basis-Szenario
2. Fehlertypen und Fehlerursachen
3. Mechanismen zur Fehlererkennung und -behebung
4. Fehlerkontrolle bei Bitfehlern
5. Fehlerkontrolle bei Paketfehlern
6. Flusskontrolle
7. Verbindungen
8. Zusammenfassung

4.2 Fehlertypen und Fehlerursachen

- Verfälschung von Bits bei der Übertragung – Bitfehler
  - Beispiel
    - Null-Bit werden durch 5 Volt repräsentiert; Eins-Bit durch 0 Volt
    - Entscheidungsschwelle sei 2,5 Volt
    - Übertragung ist nicht optimal: Rauschen, Signaldämpfung
    - Ergebnis: Empfänger empfängt Signalwert von 3 Volt, obwohl ursprünglich 0 Volt gesendet wurde
      - Es handelt sich dann um ein sogenanntes „umgekipptes“ Bit, d.h. um einen Bitfehler
  - Fehlerursachen
    - Rauschen
    - Signaldämpfung
    - Verlust der Bit-Synchronisation
    - …

Bitfehler

- Weiterhin wird unterschieden zwischen
  - Einzelbitfehler
    - Z.B. Rauschspitzen, die die Detektionsschwelle bei digitaler Signalerfassung überschreiten
    - Ein *einzelnes* Bit ist fehlerhaft
  - Bündelfehler
    - Länger anhaltende Störung durch Überspannung, Starkstromschaltprozesse etc.
    - *Mehrere direkt aufeinanderfolgende* Bits sind fehlerhaft
  - Synchronisationsfehler
    - Empfänger kann den Anfang eines Bits nicht korrekt detektieren
    - *Alle* Bits werden falsch erkannt

Fehlerhäufigkeit und Fehlerauswirkungen

- Bitfehlerrate Maß für die Fehlerhäufigkeit

$$Bitfehlerrate = \frac{Summe\ gestörter\ Bits}{Summe\ übertragener\ Bits}$$

- Typische Werte für Bitfehlerrate
  - Analoges Fernsprechnetz: $2 * 10^{-4}$
  - Funkstrecke: $10^{-3} - 10^{-4}$
  - Ethernet (10Base2): $10^{-9} - 10^{-10}$
  - Glasfaser: $10^{-10} - 10^{-12}$

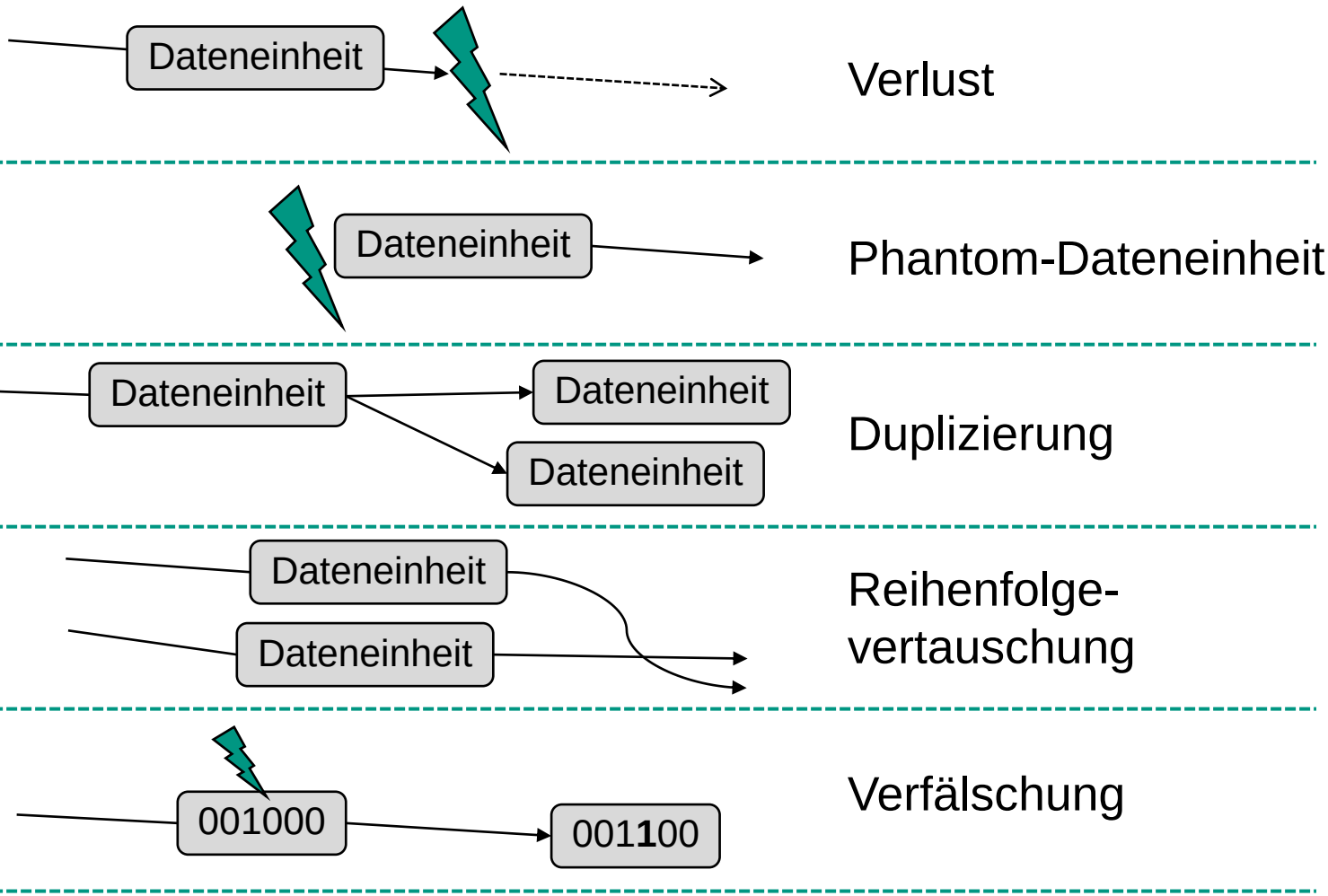
- Fehlerwirkungen gestörter Bits können sehr unterschiedlich sein
  - (Nutz-)Datenfehler
    - Bits innerhalb der Nutzdaten einer Schicht werden gestört
  - Protokollfehler
    - Störungen können Protokollkontrolldaten, Steuerzeichen, Adressen oder sonstige protokollrelevante Daten verfälschen oder vernichten

Fehlertypen und Fehlerursachen

- Verfälschung von Dateneinheiten – oft als Paketfehler bezeichnet
  - Begriff Dateneinheit häufig als Oberbegriff verwendet für
    - Rahmen, Nachrichten, Pakete, Segmente …
  - Fehlerarten
    - Verlust einer Dateneinheit
    - Empfang einer Phantom-Dateneinheit
    - Duplizierung einer Dateneinheit
    - Abweichung der Empfangsreihenfolge von Dateneinheiten
  - Fehlerursachen
    - Überlastung von Zwischensystemen
    - Unterschiedliche Wege durch das Netz
    - Verfrühte Datenwiederholung
    - …

Überblick Paket- und Bitfehler

- Paket-fehler



- Bit-fehler

4.2 Fehlerauswirkungen

- Auswirkung einer Störung u.a. abhängig von der Datenrate
  - Beispiel: Eine Störung von 20 ms führt…
    - Bei Telex (50 bit/s, Bitdauer: 20 ms)
      - zu einem Fehler von 1 Bit
      - → *Einzelbitfehler*
    - Bei ISDN (64 kbit/s, Bitdauer: 15,625 µs)
      - zu einem Fehler von 1280 Bit
      - → *Bündelfehler*
    - Bei ADSL2+ (16 Mbit/s, Bitdauer: 62,5 ns)
      - zu einem Fehler von ca. 320 kbit
      - → *Bündelfehler*

Pingo

- Pingo-Link für diese Vorlesung:

→ http://pingo.upb.de/6466

Pingo

http://pingo.upb.de/

Kapitelübersicht

1. Einführung
2. Netzwerkarchitekturen
3. Physikalische Grundlagen
4. Protokollmechanismen
5. Die Sicherungsschicht: HDLC
6. Die Sicherungsschicht: Lokale Netze
7. Netzkopplung und Vermittlung
8. Die Transportschicht
9. Sicherheit
10. Anwendungssysteme

1. Basis-Szenario
2. Fehlertypen und Fehlerursachen
3. Mechanismen zur Fehlererkennung und -behebung
4. Fehlerkontrolle bei Bitfehlern
5. Fehlerkontrolle bei Paketfehlern
6. Flusskontrolle
7. Verbindungen
8. Zusammenfassung

4.3 Kommunikation und Fehlerbehandlung

- Unzuverlässiger Kommunikationsdienst
  - Es wird keinerlei Aussage darüber getroffen, wie viel Daten beim Empfänger ankommen
    … meist geht man davon aus, dass Großteil der Daten korrekt ankommt
  - Bei Fehlern werden keine weiteren Maßnahmen unternommen

- Zuverlässiger Kommunikationsdienst
  - Übertragungsdienst garantiert für Empfänger folgendes
    - Alle Daten korrekt und vollständig
    - In der richtigen Reihenfolge
    - Ohne Duplikate
    - Ohne Phantom-Dateneinheiten
  - Bei Fehlern sind entsprechende Maßnahmen erforderlich
    - Mechanismen zur Fehlererkennung und -behebung

Beispiele verwendeter Protokollmechanismen

- Bei Bitfehlern
  - Fehlererkennende Codes
  - Fehlerkorrigierende Codes
- Bei Paketfehlern
  - Zur Erkennung erforderlich
    - Sequenznummern
    - Zeitüberwachung
    - Quittungen
  - Reaktion auf Paketfehler
    - Sendewiederholungen
  - Prävention
    - Vorwärtsfehlererkennung
      - Redundanz bereits beim Senden hinzufügen

Kapitelübersicht

1. Einführung
2. Netzwerkarchitekturen
3. Physikalische Grundlagen
4. Protokollmechanismen
5. Die Sicherungsschicht: HDLC
6. Die Sicherungsschicht: Lokale Netze
7. Netzkopplung und Vermittlung
8. Die Transportschicht
9. Sicherheit
10. Anwendungssysteme

1. Basis-Szenario
2. Fehlertypen und Fehlerursachen
3. Mechanismen zur Fehler-erkennung und -behebung
4. Fehlerkontrolle bei Bitfehlern
5. Fehlerkontrolle bei Paketfehlern
6. Flusskontrolle
7. Verbindungen
8. Zusammenfassung

4.4 Fehlerkontrolle bei Bitfehlern

- Problem
  - Wie können Bitfehler beim Empfänger oder in netzinternen Zwischensystemen erkannt werden?

- Grundlegende Ansätze
  - Ausnutzung der „Distanz“ zwischen gültigen Codewörtern,
    - nicht alle Codewörter, die mit den vorhandenen Bits erzeugt werden können, sind gültig
    - z.B. Hamming-Abstand
  - Hinzufügen von Redundanz bei der Übertragung
    - z.B. Paritätsbits, Prüfsumme …

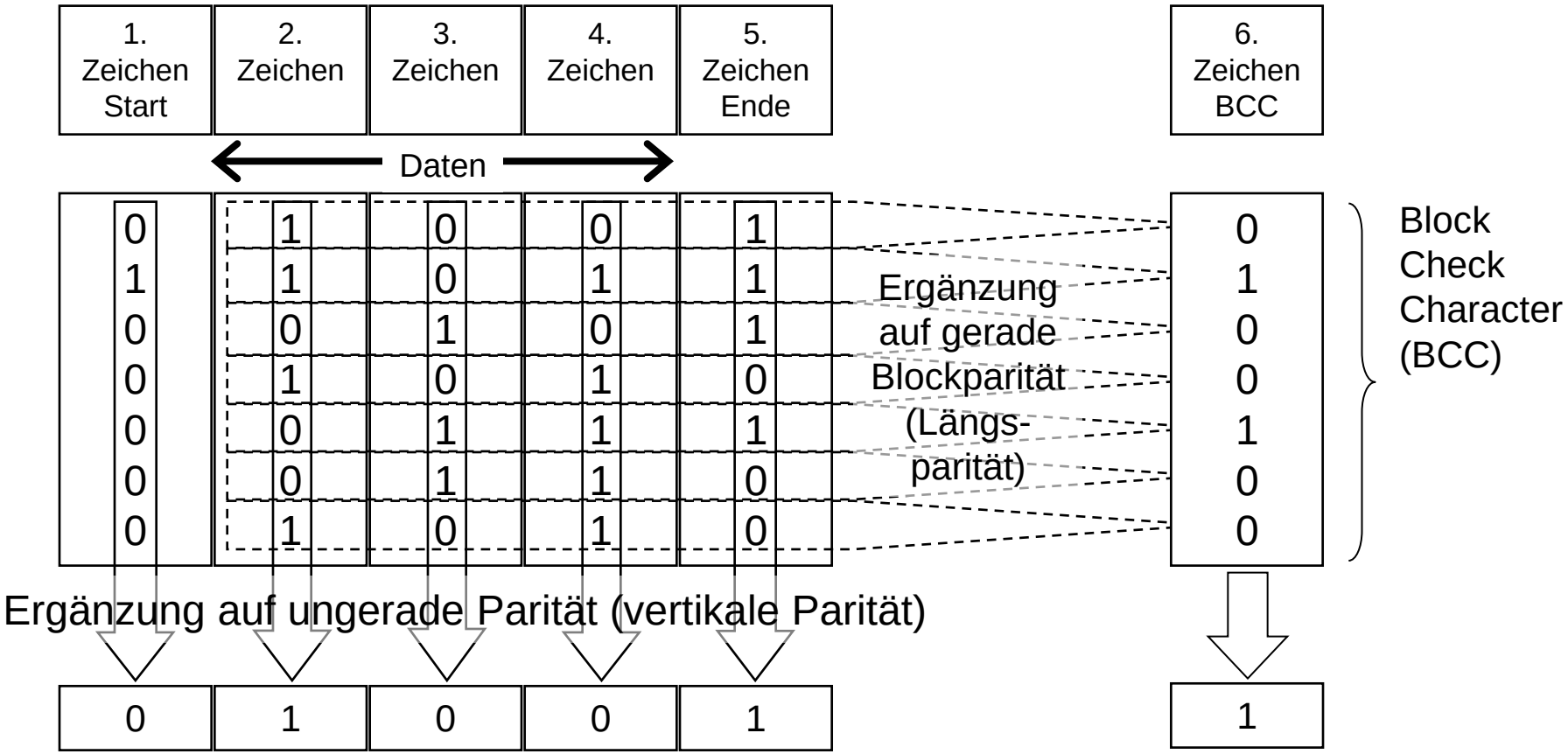
4.4.1 Paritätsbits

- Zu einer „Einheit“ wird ein redundantes Bit hinzugefügt
  - Gerade Parität
    - Es wird auf gerade Anzahl von 1-Bits ergänzt
  - Ungerade Parität
    - Es wird auf ungerade Anzahl von 1-Bits ergänzt

- Folgende Varianten werden unterschieden
  - Vertikale Parität
    - An jedes Zeichen (bestehend aus n Bits) wird ein Paritätsbit angefügt
      - d.h. ein Paritätsbit pro Spalte
    - Erkennung von Bitfehlern ungerader Anzahl (1-Bitfehler, 3-Bitfehler etc.)
  - Längsparität
    - An Folge von Zeichen wird dediziertes Prüfzeichen angefügt
      - Enthält ein Paritätsbit pro Reihe (d.h. pro n-tem Bit aller Zeichen)
    - Auch als *Block Check Character* (BCC) bezeichnet

Paritätsbits: Beispiel

- Paritätssicherung bei Zeichen-basierter Übertragung
  - Modulo-2-Arithmetik



Einschub: Code

- Gegeben seien ein Alphabet A und ein Alphabet B
  - Code: injektive Abbildung $f: A \rightarrow B^*$
  - Codewort
    - Für $a \in A$ heißt $f(a)$ ein Codewort von f
  - Beispiele
    - Morse Code
    - ASCII Code
    - …

Morse-Alphabet					
A	.–	N	–.	0	–––––
B	–...	O	–––	1	.––––
C	–.–.	P	.––.	2	..–––
D	–..	Q	––.–	3	...––
E	.	R	.–.	4	....–
F	..–.	S	...	5	.....
G	––.	T	–	6	–....
H	....	U	..–	7	––...
I	..	V	...–	8	–––..
J	.–––	W	.––	9	––––.
K	–.–	X	–..–	Punkt	.–.–.–
L	.–..	Y	–.––	Komma	––..––
M	––	Z	––..	?	..––..

4.4.2 Hamming-Abstand

- Hamming-Abstand d einzelner Codewörter
  - Anzahl der Bitpositionen, in denen sich zwei Codewörter $c_1$und c_2 unterscheiden
  - Beispiel
    - $d(10001001, 10110001) = 3$ (Anzahl der Eins-Bits von c_1 XOR c_2)

- Hamming-Abstand D des vollständigen Codes C

$$D(C) := \min\{d(c_1, c_2) \mid c_1, c_2 \in C, c_1 \neq c_2\}$$

- Hamming-Abstand bestimmt die Fähigkeit eines Codes, Fehler zu erkennen und zu beheben
  - Erkenne k-Bitfehler: Hamming-Abstand von $k + 1$ notwendig
  - Behebe k-Bitfehler: Hamming-Abstand von $2k + 1$ notwendig

Hamming-Abstand

- Codewort besteht aus m Bits
  - Welche Anzahl Prüfbits r werden benötigt, um *1-Bitfehler* zu beheben?

m	r?

$$n = m + r$$

  - 2^m legale Codewörter können mit m Bits erzeugt werden
  - Allgemein gilt: Pro Codewort c_i der Länge $n = m + r$ existieren n illegale Codewörter gleicher Länge mit Hamming-Abstand 1
    - Jeweils mit einem komplementierten Bit
    - Nur Codewort c_i selbst hat Hamming-Abstand 0
  - Es soll 2^m legale Codewörter im Code geben, von denen jedes $n + 1$ Codewörter belegt (n illegale und ein legales, c_i selbst)
  - Hierzu müssen $(n + 1)2^m$ Codewörter mit $n = m + r$ Bits darstellbar sein
    - $(n + 1)2^m \leq 2^n \rightarrow (m + r + 1) \leq 2^r$ untere Grenze für r

Beispiele

- Fehlererkennender Code
  - Code mit einem einzigen Paritätsbit (gerade oder ungerade)
  - Erkennen eines 1-Bitfehlers möglich (oder aller Fehler mit einer ungeraden Anzahl Bits)

- Fehlerbehebender Code
  - Beispiel: `00000 00000,`
    `00000 11111,`
    `11111 00000,`
    `11111 11111`
  - Hamming-Abstand = 5
  - Korrektur von 2-Bitfehlern möglich
  - Beispiel: `00000 00111` → `00000 11111`

4.4.3 Prüfsumme

- An Dateneinheit wird eine Sicherungssequenz angefügt
  - Auch als FCS (*Frame Check Sequence*) bezeichnet

- Cyclic Redundancy Check (CRC)
  - Basiert auf Division in Modulo-2-Arithmetik; keine Überträge
  - Entspricht bitweiser **XOR**-Operation
    - `1+1 = 0+0 = 0, 1+0 = 0+1 = 1`
  - Beispiele für bitweise **XOR**-Operation

```
 10011011      00110011      11110000      01010101
+11001010     +11001101     -10100110     -10101111
----------    ----------    ----------    ----------
 01010001      11111110      01010110      11111010
```

- Bitstrings als Repräsentation von Polynomen
  - Dateneinheit wird als unstrukturierte Bitfolge aufgefasst
  - Dateneinheit **`10011010`** entspricht Polynom $M(x) = x^7 + x^4 + x^3 + x^1$

Vorgehensweise

- Gleiches Generatorpolynom $G(x)$ für Sender und Empfänger
  - Höchstes und niederwertigstes Bit von $G(x)$ müssen 1 sein
- Prüfsumme (*Checksum*) wird berechnet
  - Dateneinheit mit m Bits entspricht $M(x)$
  - Prüfsumme entspricht Rest R der Division $(x^r\ M(x))/G(x)$
    - r: Grad des Generatorpolynoms; $m > r$
    - $x^r\ M(x)$ fügt r Nullstellen an das Ende der Dateneinheit
- Prüfsumme wird an die zu sendenden Daten angehängt
  - Entspricht der Addition des Restes: $x^r\ M(x) + R$
- Empfänger überprüft Dateneinheit
  - Division der empfangenen Dateneinheit durch $G(x)$
    - Ist der Rest der Division Null, dann wurde kein Fehler erkannt
    - Ist der Rest ungleich Null, dann ist die empfangene Dateneinheit fehlerhaft

CRC: Beispiel

- Generatorpolynom **G(x) = 1101**; Nachricht **M(x) = 1001 1010**

```
1001 1010 000     /    1101    =   1111 1001
------------------------------
1101
-------
 100 1
 110 1
 --------
  10 00
  11 01
  --------
   1 011
   1 101
   --------
     1100
     1101
     -------
        1 000
        1 101
        --------
          101
```

- Welche Bitfolge wird übertragen?

Erkennen von Bitfehlern mit CRC

- Erkennen aller Einzelbitfehler
  - x^r und x^0 des Generatorpolynoms dürfen nicht gleich Null sein
- Nahezu alle Doppelbitfehler
  - $G(x)$ muss mindestens drei Terme besitzen
  - Sämtliche Doppelbitfehler
    - $(x^k + 1)$ nicht durch $G(x)$ teilbar ($\forall k \leq$ Länge der Dateneinheit)
- Jede ungerade Anzahl an Bitfehlern
  - $G(x)$ muss den Faktor $x + 1$ enthalten
- Alle Bursts mit bis zu m Bitfehlern
  - $G(x)$ hat den Grad m

Bekannte Generatorpolynome

- International genormt sind u.a. folgende Generatorpolynome
  - CRC-12 $= x^{12} + x^{11} + x^3 + x^2 + x + 1$
  - CRC-16 $= x^{16} + x^{15} + x^2 + 1$
  - CRC-CCITT $= x^{16} + x^{12} + x^5 + 1$
  - CRC-32 $= x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

- CRC-16 und CRC-CCITT entdecken
  - alle Einzel- und Doppelfehler
  - alle Fehler ungerader Anzahl
  - alle Fehlerbursts mit der Länge ≤ 16
  - 99,997 % aller Fehlerbursts mit der Länge 17 (gilt nicht für CRC-16)
  - 99,998 % aller Fehlerbursts mit der Länge 18 und mehr (gilt nicht für CRC-16)

CRC-Implementierung in Hardware

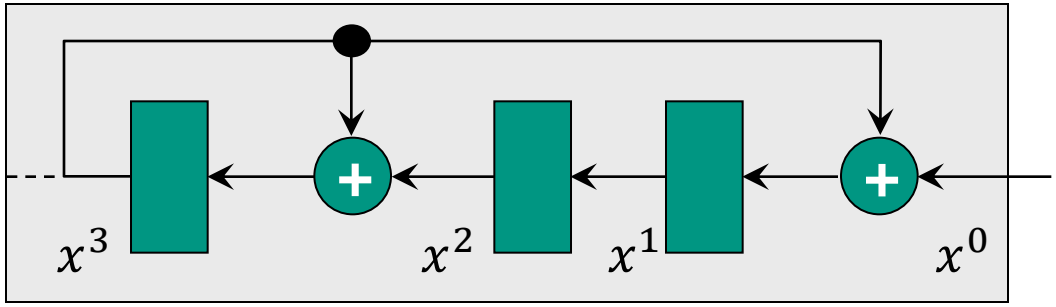
- Realisierung in Hardware
  - Benutzung von rückgekoppelten Schieberegistern
  - CRC kann während des „Durchschiebens“ durch das Schieberegister berechnet werden

- Prinzip
  - Daten werden bitweise empfangen und durchlaufen das Schieberegister
  - Rückkopplung durch ein XOR-Gatter erfolgt an den Stellen, an denen Bits im Generatorpolynom auf 1 gesetzt sind
    - Ohne das höchste Bit, dort erfolgt die Rückkopplung
  - Nach Durchschieben der Dateneinheit und ihrer angehängten Nullen steht Prüfsumme im Register
    - Zu beachten: Soll Prüfsumme ebenfalls „aus dem Register geschoben werden“, ist noch etwas mehr Schaltungslogik nötig

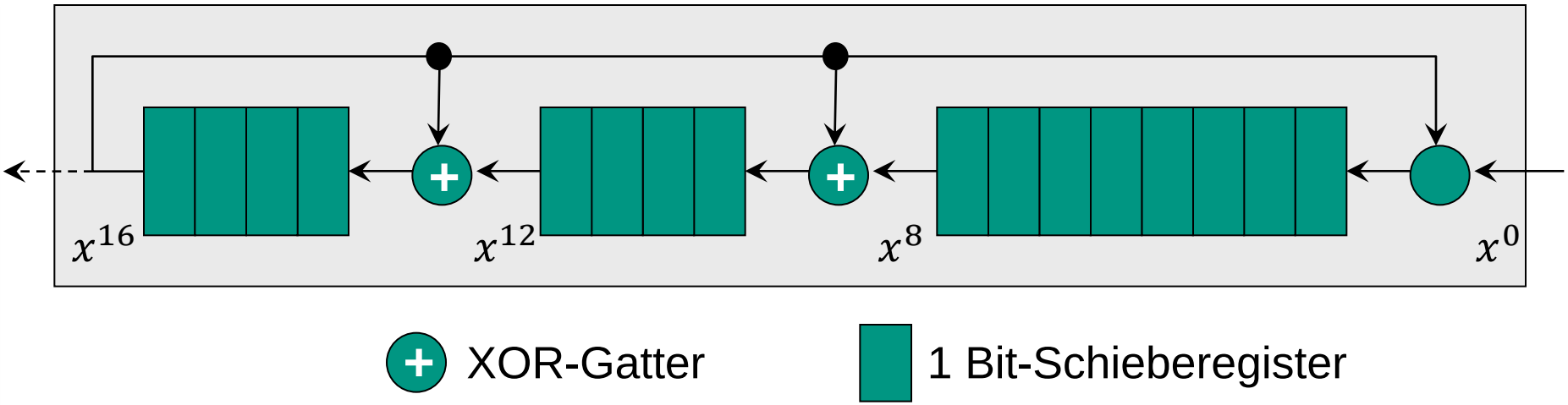
Beispiele Schieberegister

- $G(x) = 1101$

$$G(x) = x^3 + x^2 + 1$$



- $G(x) = x^{16} + x^{12} + x^8 + 1$



4.4.4 Internet-Prüfsumme

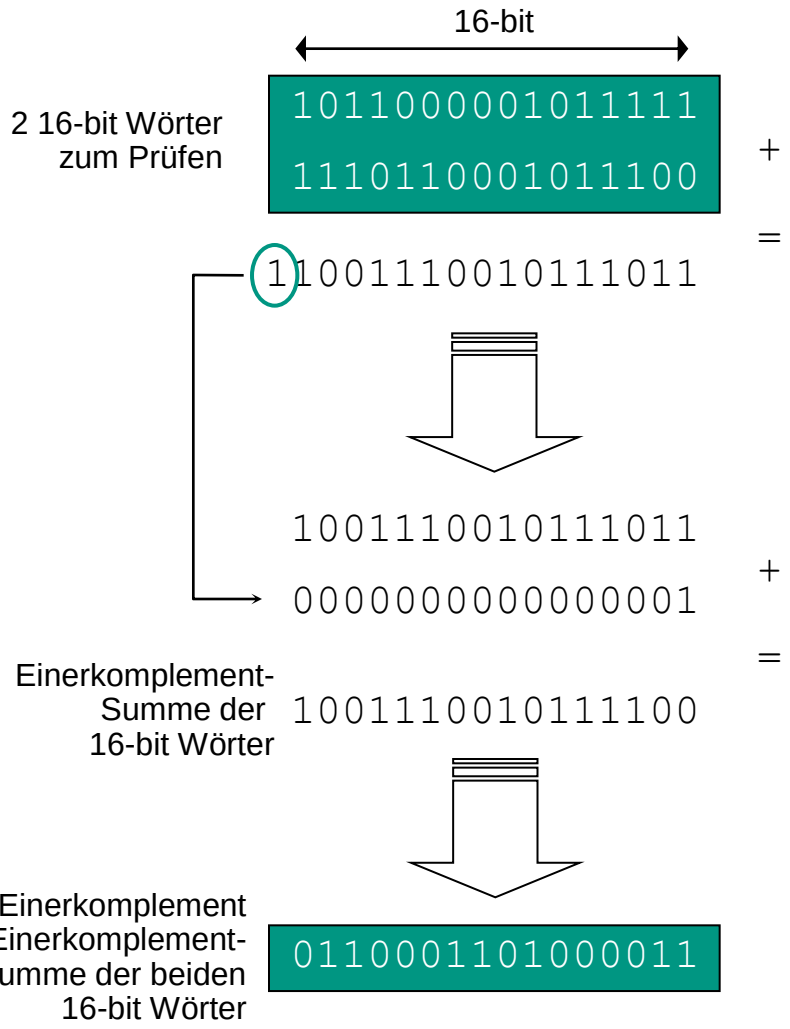
- Internet-Protokolle verwenden andere Variante für Prüfsummen
  - Speziell für Realisierung in Software ausgelegt
  - In den Eigenschaften zur Fehlererkennung nicht so gut wie CRC
    - Wörter in falscher Reihenfolge können nicht erkannt werden
  - Verwendet bei: Internet Protocol (IP), User Datagram Protocol (UDP), Transmission Control Protocol (TCP)

- Prinzip
  - Aufaddieren aller übertragenen Wörter (16 Bit Wortlänge)
    - Wörter werden als **`Integer`** aufgefasst
    - $Prüfsumme = \sum alle\ übertragenen\ Wörter$
  - Prüfsumme wird mit Dateneinheit übertragen

- Implementierung
  - Addition unter Verwendung des Einer-Komplements
  - RFC 1071: Hinweise und Techniken für effiziente Implementierungen

Beispiel IPv4

- Prüfsumme für Protokollkopf
  - In jedem Router neu berechnet, da sich Werte im Protokollkopf ändern können (z.B. Time to Live)
- Prüfsummenberechnung
  - Addition aller 16-bit Wörter
    - Übertrag auf Least Significant Bit addieren
  - Einerkomplement der Summe



[Benv05]

Kapitelübersicht

1. Einführung
2. Netzwerkarchitekturen
3. Physikalische Grundlagen
4. Protokollmechanismen
5. Die Sicherungsschicht: HDLC
6. Die Sicherungsschicht: Lokale Netze
7. Netzkopplung und Vermittlung
8. Die Transportschicht
9. Sicherheit
10. Anwendungssysteme

1. Basis-Szenario
2. Fehlertypen und Fehlerursachen
3. Mechanismen zur Fehler-erkennung und -behebung
4. Fehlerkontrolle bei Bitfehlern
5. Fehlerkontrolle bei Paketfehlern
6. Flusskontrolle
7. Verbindungen
8. Zusammenfassung

4.5 Fehlerkontrolle bei Paketfehlern

- Mit Paritätsbits und Prüfsummen können Bitfehler erkannt werden, allerdings nur dann, wenn die Dateneinheit beim Empfänger ankommt oder in den netzinternen Zwischensystemen analysiert wird

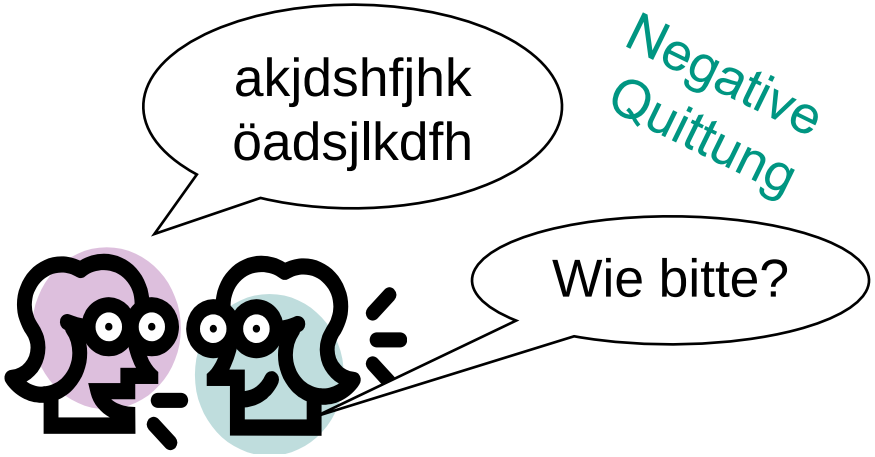
- Zur Erkennung von Fehlern bzgl. kompletter Dateneinheiten (Paketfehler) sind zusätzliche Mechanismen erforderlich
  - Sequenznummern (*Sequence Number*)
  - Zeitgeber (*Timer*)

- Zur Behebung der Fehler werden die folgenden Mechanismen verwendet
  - Quittungen (*Acknowledgements*)
  - Sendewiederholungen (*Retransmissions*)

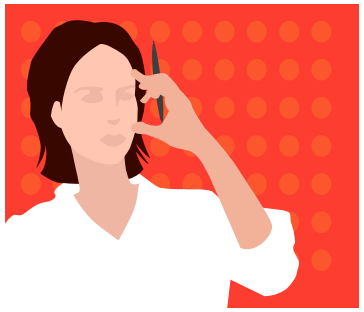
… Verständigungsprobleme?

Sendewiederholung

Negative Quittung



Positive Quittung



Sequenz-nummern

1, 2, 3, 5, … etwas fehlt hier …

4.5.1 Sequenznummern

- Problem
  - Woher weiß der Empfänger, ob
    - Dateneinheiten in der richtigen Reihenfolge ankommen?
    - keine Duplikate enthalten sind?
    - keine Dateneinheiten fehlen?
- Mechanismus
  - Dateneinheiten (oder die Bytes) werden durchnummeriert
  - Entsprechende Kennung wird mit jeder Dateneinheit übertragen
  - Kennung wird als Sequenznummer bezeichnet
- Fehlerszenarien
  - Sequenznummern helfen, wenn Dateneinheiten nicht ausgeliefert werden
    - Beispiele?
- Größe
  - Bei einer Länge der Sequenznummer von n Bit umfasst der Sequenznummernraum 2^n Sequenznummern

4.5.2 Quittungen

- Problem
  - Wie erfährt der Sender, dass eine Dateneinheit überhaupt nicht bzw. nicht korrekt beim Empfänger angekommen ist?
- Mechanismus
  - Empfänger informiert Sender, ob er Dateneinheit empfangen hat oder nicht
  - Versendung spezieller Dateneinheiten, sogenannte Quittungen (ACK: Acknowledgement)
- Varianten
  - Positive Quittung
    - Empfänger teilt dem Sender mit, dass er die Daten erhalten hat
  - Negative Quittung
    - Empfänger meldet dem Sender, dass er die Daten nicht erhalten hat (z.B. wenn er nachfolgende Dateneinheiten erhält)
    - NACK: Negative Acknowledgement

Quittungen

- Weitere Varianten
  - Selektive Quittungen
    - SACK: Selective Acknowledgement
    - Quittung bezieht sich auf eine einzelne Dateneinheit
    - Beispiel
      - Negative selektive Quittung, falls der Verlust einer Dateneinheit vom Empfänger vermutet wird (NACK)
  - Kumulative Quittungen
    - Quittung bezieht sich auf eine Menge von Dateneinheiten, die in der Regel durch eine obere Sequenznummer beschränkt ist
    - Beispiel
      - Positive kumulative Quittung, die besagt, dass alle Dateneinheiten bis zur angegebenen Sequenznummer korrekt empfangen wurden
- Quittungen werden oftmals in Kombination mit Zeitgebern verwendet

Pingo

http://pingo.upb.de/

4.5.3 Zeitgeber

- Problem
  - Woran merkt ein Sender, dass eine Dateneinheit nicht angekommen ist?
- Mechanismus
  - In Abhängigkeit einer zeitlichen Obergrenze wird vermutet, dass eine Dateneinheit beim Empfänger nicht angekommen ist
  - Sender kann dann Sendewiederholung starten
- Implementierung
  - Welcher Wert wird für den Zeitgeber gewählt?

4.5.4 Automatic Repeat Request: ARQ

- Grundlegende Variante zur Sendewiederholung
  - Sender erhält positive Quittungen über den Erhalt einer Dateneinheit vom Empfänger
  - Sender kann Sendewiederholungen ausführen

- Varianten
  - Wann werden Quittungen versendet?
  - Wann werden Sendewiederholungen veranlasst?

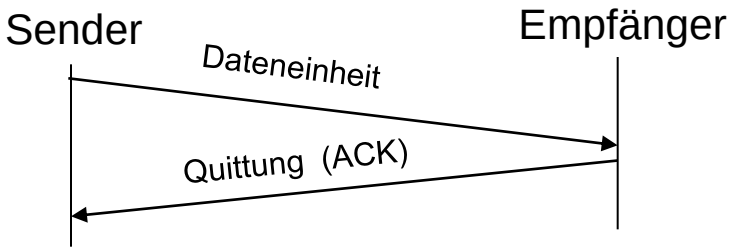
Stop-and-Wait

- Sehr einfaches ARQ-Verfahren
  - Sender wartet auf Quittung zu einer gesendeten Dateneinheit, bevor er neue Dateneinheit senden darf
  - Falls keine Quittung empfangen wird, erfolgt Wiederholung der Dateneinheit
  - Wartezeit auf Quittung wird durch Zeitgeber geregelt

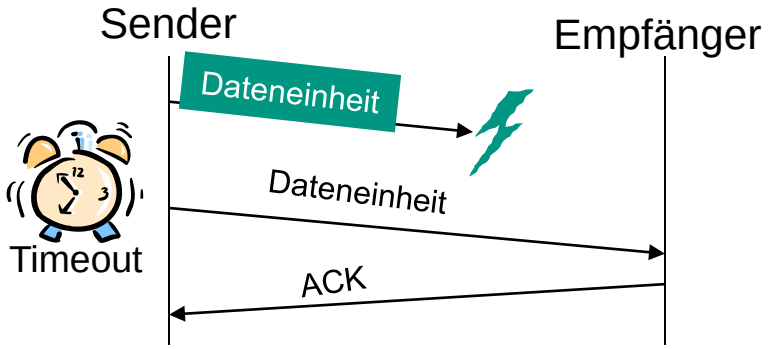
- Einsatzbeispiel
  - WLAN

Stop-and-Wait: Szenarien

Regulärer Ablauf

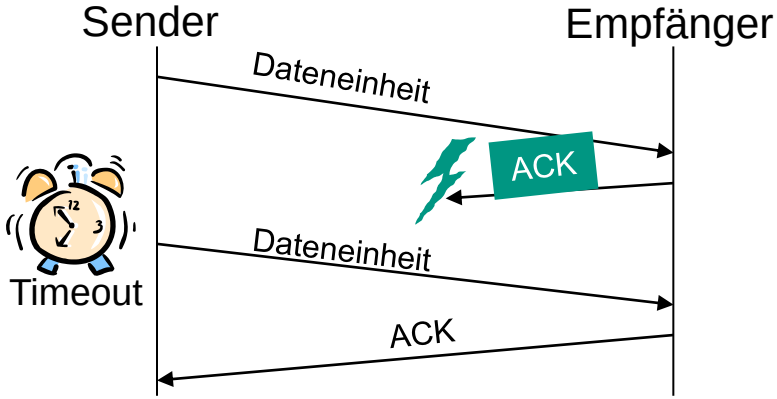


Verlust einer Dateneinheit



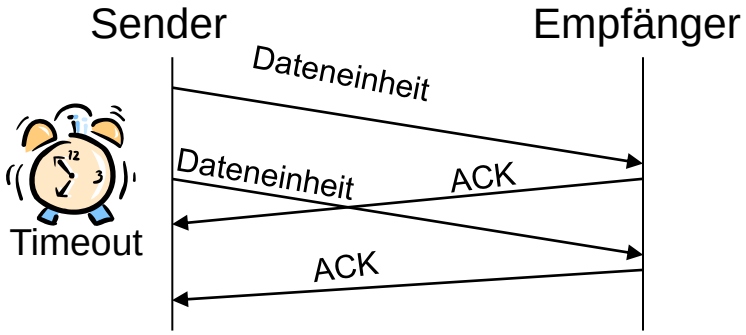
Verlust einer Quittung

- Unterschied zum Verlust einer Dateneinheit?



Zu schneller Ablauf des Zeitgebers

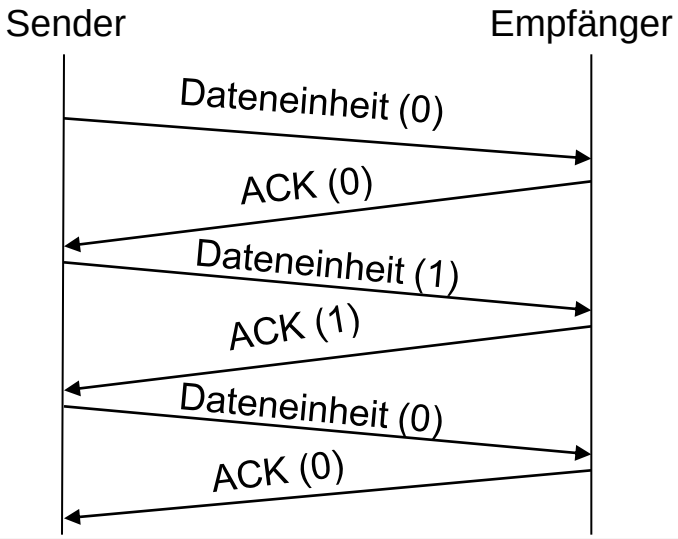
- Problem?



- Frage: Auf welchen Wert wird der Zeitgeber gesetzt?
- Frage: Unterscheiden sich Verluste bzw. Übertragungsfehler in der Behandlung?

Stop-and-Wait: Sequenznummern

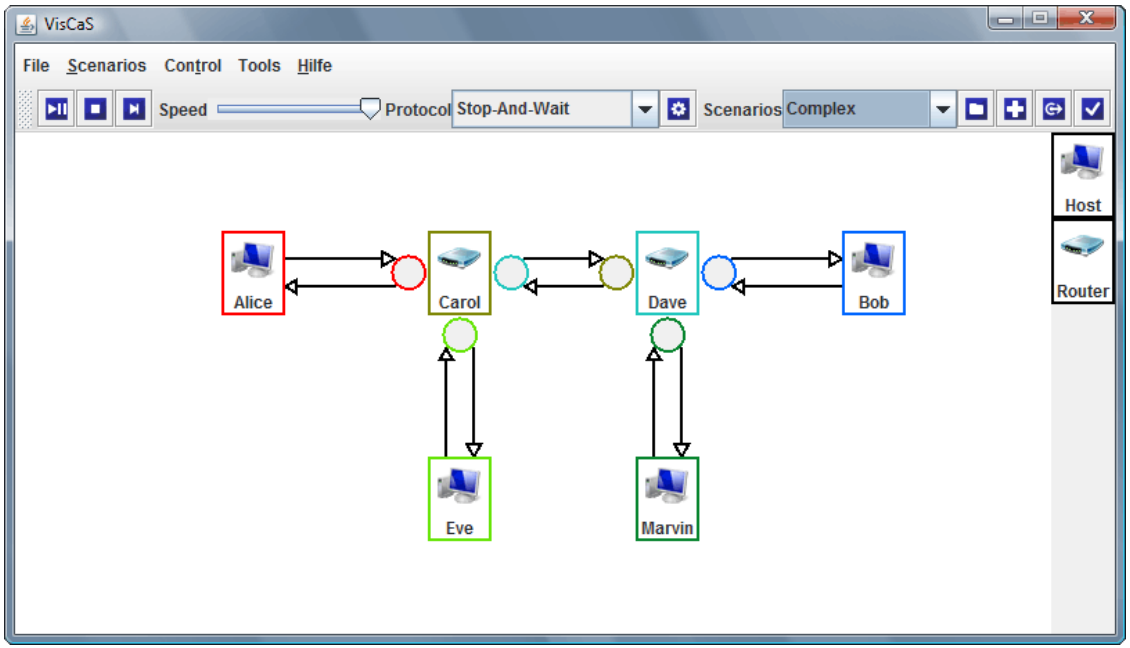
- Problem
  - In den vorangegangenen Szenarien besteht die Möglichkeit, dass der Empfänger eine Dateneinheit doppelt erhält
    - Er kann dies nicht erkennen
- Mechanismus
  - Sequenznummern
    - Die Dateneinheiten werden mit einer Kennung versehen, die es dem Empfänger ermöglicht, diese zu unterscheiden
    - Für Stop-and-Wait ist eine Sequenznummer von einem Bit ausreichend (0 und 1)
- Ablauf



Java-Anwendung: Stop-and-Wait

Testen Sie unsere Java-Anwendung und Lernumgebung im Internet:
http://www.tm.kit.edu/software/viscas/

- Einfaches Szenario mit zwei direkt verknüpften Rechnern
- Testen Sie die Leistungsfähigkeit der Fehlerkontrolle
  - Experimentieren Sie mit unterschiedlichen Verlust- bzw. Fehlerraten
  - Experimentieren Sie mit unterschiedlichen Datenraten



Leistungsbewertung

- Nachteil Stop-and-Wait
  - Sender darf nur eine Dateneinheit senden und muss dann auf Quittung warten

→ Wie sieht es mit der erzielbaren Leistungsfähigkeit aus?

- Kriterien zur Leistungsbewertung
  - Durchsatz (engl. *Throughput*)
    - Maß für die Menge an Daten, die pro Zeiteinheit übertragen werden kann
  - Auslastung (engl. *Utilization* – U)
    - Maß für die Auslastung einer Ressource (z.B. Übertragungsmedium)
    - Verhältnis von tatsächlicher Nutzung zu möglicher Nutzung
      - Tatsächliche Nutzung: erfolgreich übertragene Daten(einheiten)
      - Mögliche Nutzung: Wie viele Daten(einheiten) hätten in dieser Zeit übertragen werden können?

Leistungsparameter

- Durchsatz
  - Synonym: Datenrate
  - Gemessen in bit/s

  … kann an unterschiedlichen Stellen eines Kommunikationssystems angegeben / gemessen werden

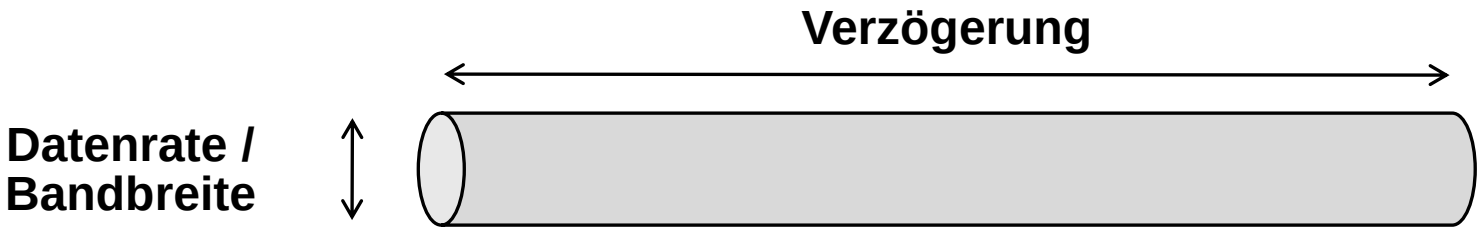
Anmerkung: Begriff Bandbreite manchmal als Synonym verwendet. Bezeichnet aber auch die Breite eines Frequenzbands

Leistungsparameter

- Verzögerung / Latenz
  - Verarbeitungsverzögerung (engl. *Processing Delay*)
    - Zeit, die nötig ist um eine Dateneinheit zu verarbeiten
      - z.B. Berechnung von Prüfsummen
  - Warteschlangenverzögerung (engl. *Queuing Delay*)
    - Zeit, die gewartet werden muss bis eine Dateneinheit gesendet werden kann
    - abhängig von der Auslastung des Netzes
  - Sendezeit (engl. *Transmission Delay*)
    - Zeit, die nötig ist um n Bits zu senden
    - abhängig von der Datenrate
  - Ausbreitungsverzögerung (engl. *Propagation Delay*)
    - Zeit, die ein Bit von A nach B benötigt
    - abhängig von der Länge des Mediums und der Ausbreitungsgeschwindigkeit

Bandbreiten-Verzögerungs-Produkt

- *Bandbreite * Verzögerung*
- *Bandbreite * RTT / 2*



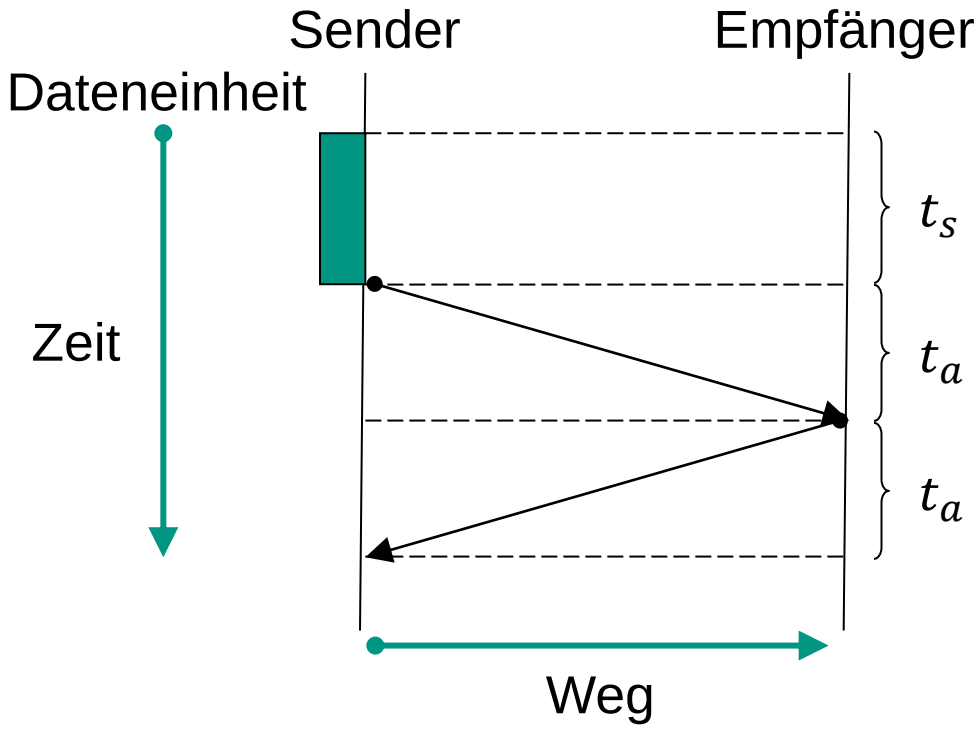
Anschluss	Datenrate (typisch)	Entfernung (typisch)	Round-Trip-Time	Bandbreiten-Verzögerungs-Produkt
Modem	56 kbit/s	10 km	87 µs	2,5 bits
WLAN	54 Mbit/s	50 m	0,33 µs	9 bits
Satellit	45 Mbit/s	35.786 km	230 ms	5,4 Mbit
Überland-Glasfaser	10 Gbit/s	4.000 km	40 ms	200 Mbit

Stop-and-Wait: Leistungsbewertung

- Annahme: Fehlerfreie Kommunikation
- Welche Parameter haben einen Einfluss?
  - Wie schnell ist eine komplette Dateneinheit bzw. eine Quittung auf das Medium gesendet?
    - Sendezeit t_s
      - t_s = Länge Dateneinheit / Datenrate
  - Wie lange benötigt ein Bit vom Sender zum Empfänger (und umgekehrt)?
    - Ausbreitungsverzögerung t_a
      - t_a = Länge des Mediums / Ausbreitungsgeschwindigkeit
  - Wie viel Zeit wird für die Verarbeitung einer Dateneinheit bzw. einer Quittung benötigt?
    - Verarbeitungszeit t_V
- Vereinfachungen
  - Verarbeitungszeit der Dateneinheit wird vernachlässigt
  - Sende- und Verarbeitungszeit einer Quittung werden vernachlässigt
    - Quittung klein; Verarbeitung schnell

Stop-and-Wait: Gesamtübertragungsdauer

- Insgesamt benötigte Zeit t_{Ges}
  - $t_{Ges} = t_s(\text{Daten}) + t_a + t_V(\text{Daten}) + t_s(\text{Quittung}) + t_a + t_V(\text{Quittung})$
- Mit Vereinfachungen
  - $t_{Ges} = t_s(\text{Daten}) + 2t_a = t_s + 2t_a$



Stop-and-Wait: Auslastung des Mediums

- Auslastung U
  - Tatsächliche Nutzung
    - Sendezeit der Dateneinheit: t_s
  - Mögliche Nutzung
    - Zeitintervall vom Beginn des Sendens der Dateneinheit bis zum vollständigen Empfang der Quittung: t_{Ges}

$$U = \frac{t_s}{t_{Ges}} = \frac{t_s}{t_s + 2t_a} \qquad \text{bzw.} \quad U = \frac{1}{1 + 2t_a/t_s}$$

- Mit $a = t_a/t_s$ ergibt sich

$$U = \frac{1}{1 + 2a}$$

Parameter a

- Definition
  - $a = t_a / t_s$ = Ausbreitungsverzögerung / Sendezeit
- Einflussgrößen
  - Länge des Mediums: m
  - Ausbreitungsgeschwindigkeit: v
  - Länge der Dateneinheit: X
  - Datenrate: r

$$\rightarrow \quad a = \frac{\frac{m}{v}}{\frac{X}{r}} = \frac{\frac{m}{v} r}{X}$$

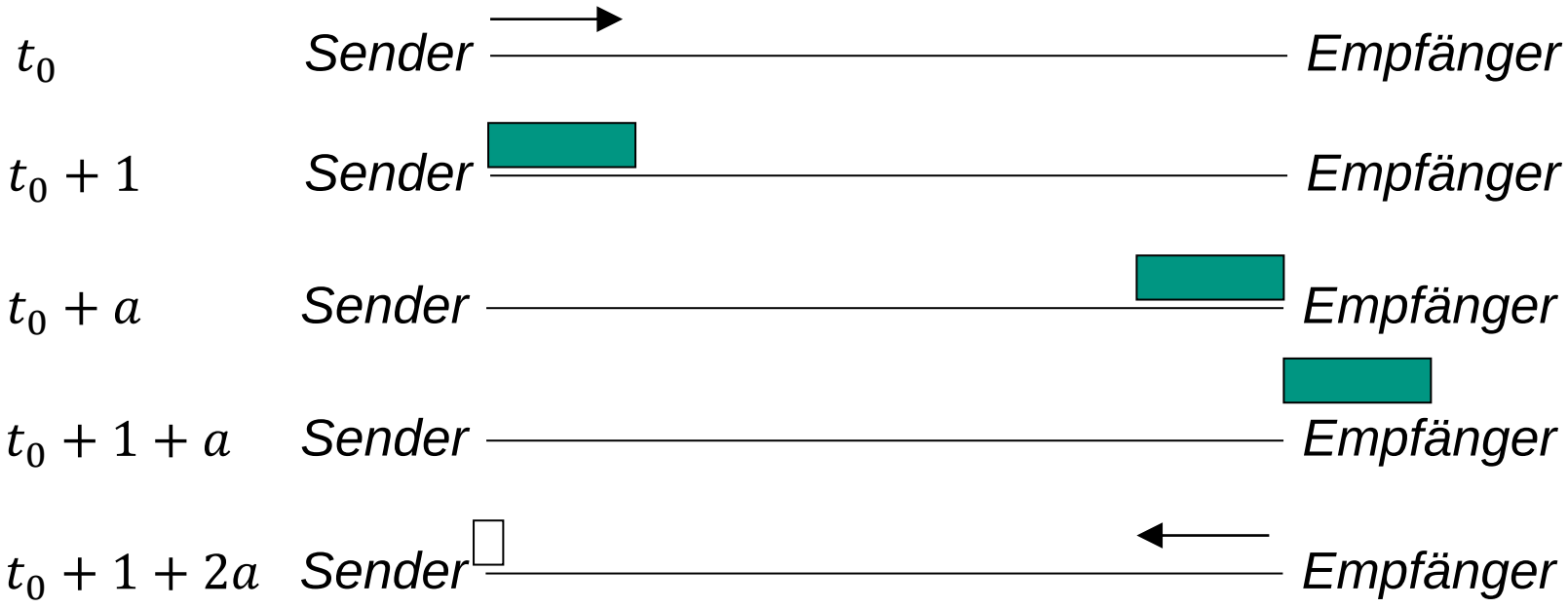
Bandbreiten-Verzögerungs-Produkt

- Interpretation
  - Bandbreiten-Verzögerungs-Produkt $(m/v * r)$: Länge des Mediums in Bit

  - Parameter a repräsentiert das Verhältnis der Länge des Mediums in Bit zur Länge der Dateneinheit

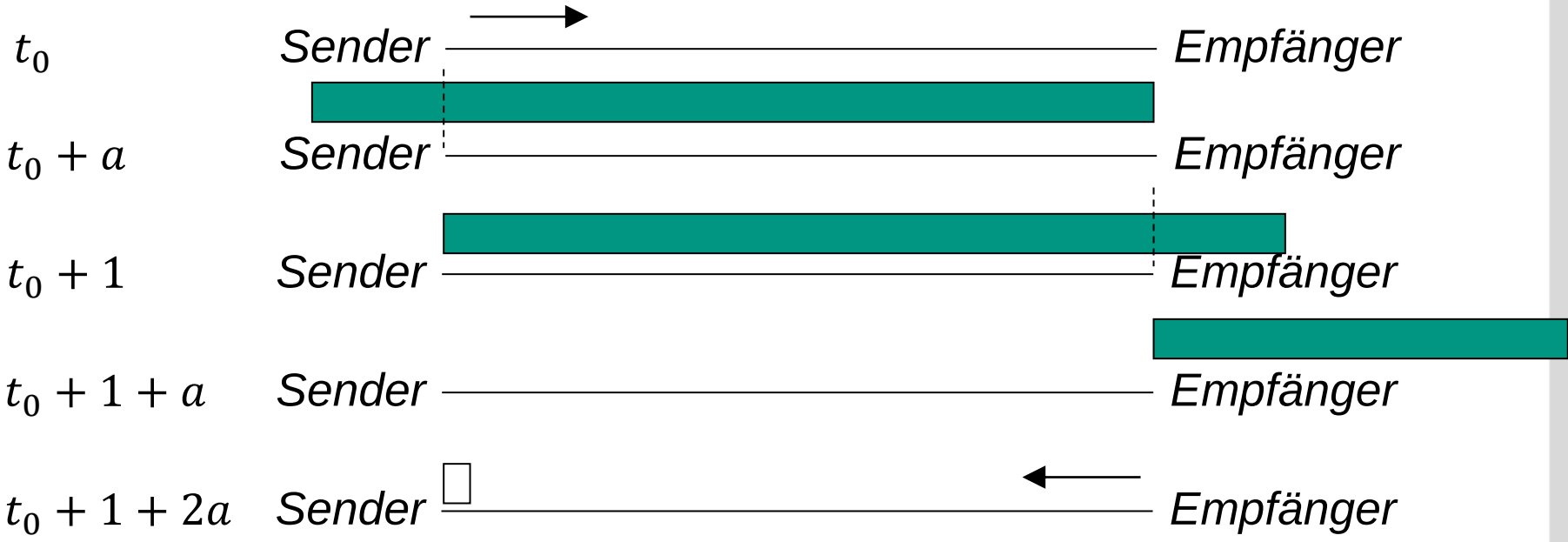
Beispiel zu Parameter a

- Normalisierung
  - Annahme: Sendezeit einer Dateneinheit betrage 1
- Damit gilt
  - $a = t_a$
  - Für $a > 1$: Ausbreitungsverzögerung > Sendezeit
    d.h. Dateneinheit passt vollständig auf das Medium



Beispiel zu Parameter a

- Für $a < 1$: Ausbreitungsverzögerung < Sendezeit
  d.h. Dateneinheit passt *nicht* vollständig auf das Medium



Stop-and-Wait: Auslastung des Mediums

- Beispiel
  - Dateneinheiten der Länge $1000\ \text{Bit}$
  - Datenraten von $1\ \text{kbit/s}$ und $1\ \text{Mbit/s}$
  - Fehlerrate vernachlässigbar

- Drei unterschiedliche Medientypen
  - Verdrilltes Adernpaar (Ausbreitungsgeschwindigkeit $2 * 10^8\ \text{m/s}$), Länge des Mediums $1\ \text{km}$
    - Auslastung?
  - Standleitung (Ausbreitungsgeschwindigkeit $2 * 10^8\ \text{m/s}$), Länge des Mediums $200\ \text{km}$
    - Auslastung?
  - Satelliten-Verbindung (Ausbreitungsgeschwindigkeit $3 * 10^8\ \text{m/s}$), Länge des Mediums $50.000\ \text{km}$
    - Auslastung?

Stop-and-Wait: Leistungsbewertung

- Jetzt
  - Berücksichtigung von Übertragungsfehlern
- Fehlerfall: Sender erhält keine korrekte Quittung
- Annahme: $n-1$ konsekutive Sendewiederholungen
  - Benötigte Zeit hierfür
    - $t_{Ges} = t_s + (n-1)(\text{Timeout} + t_s) + 2t_a$
  - Annahme
    - Timeout entspricht zweifacher Ausbreitungsverzögerung t_a
    - Dann gilt: $t_{Ges} = n(t_s + 2t_a)$
  - Damit gilt für die Auslastung

$$U = \frac{t_s}{t_{Ges}} = \frac{t_s}{n(t_s + 2t_a)}$$

  - … in der Regel n nicht fest: Erwartungswert?

Stop-and-Wait: Leistungsbewertung

- Annahmen
  - p sei Wahrscheinlichkeit, dass eine Dateneinheit fehlerhaft übertragen wird
  - Quittungen seien nie verfälscht

- Wahrscheinlichkeit für genau k Übertragungsversuche?
  - $k-1$ fehlerhafte Übertragungsversuche gefolgt von einer erfolgreichen Übertragung
    - Wahrscheinlichkeit hierfür: $p^{k-1}(1-p)$

  - $n =$ Erwartungswert [Anzahl Übertragungen]

$$n = \sum_{i=1}^{\infty} (i \times P[\text{genau } i \text{ Übertragungen}]) = \sum_{i=1}^{\infty} i p^{i-1}(1-p) = \frac{1}{1-p}$$

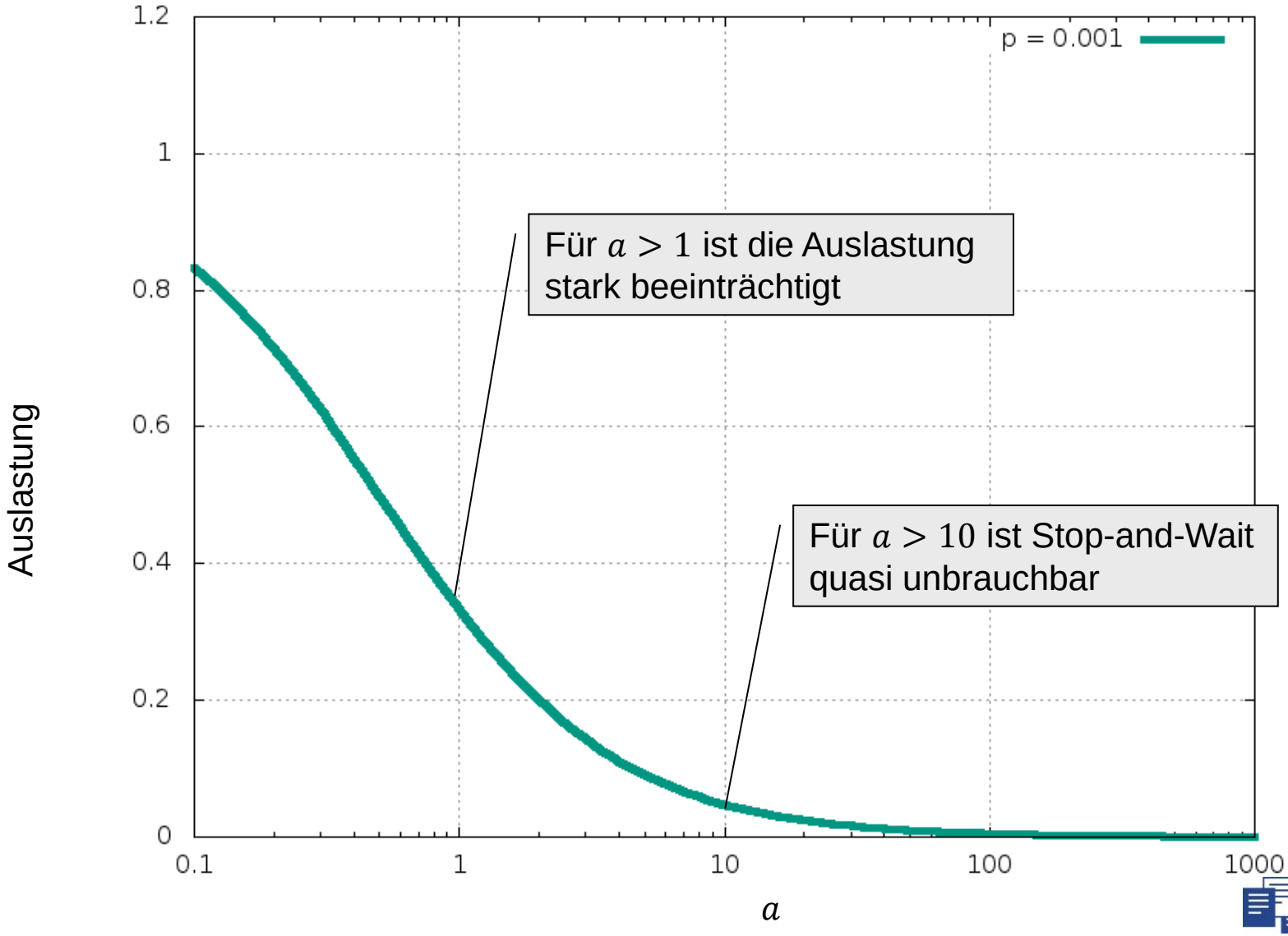
für $-1 < x < 1$ gilt:

$$\sum_{i=1}^{\infty} i x^{i-1} = \frac{1}{(1-x)^2}$$

  - Damit gilt für die Auslastung

$$U = \frac{1}{n(1+2a)} = \frac{1-p}{1+2a}$$

Bewertung von Stop-and-Wait



[Stal06]

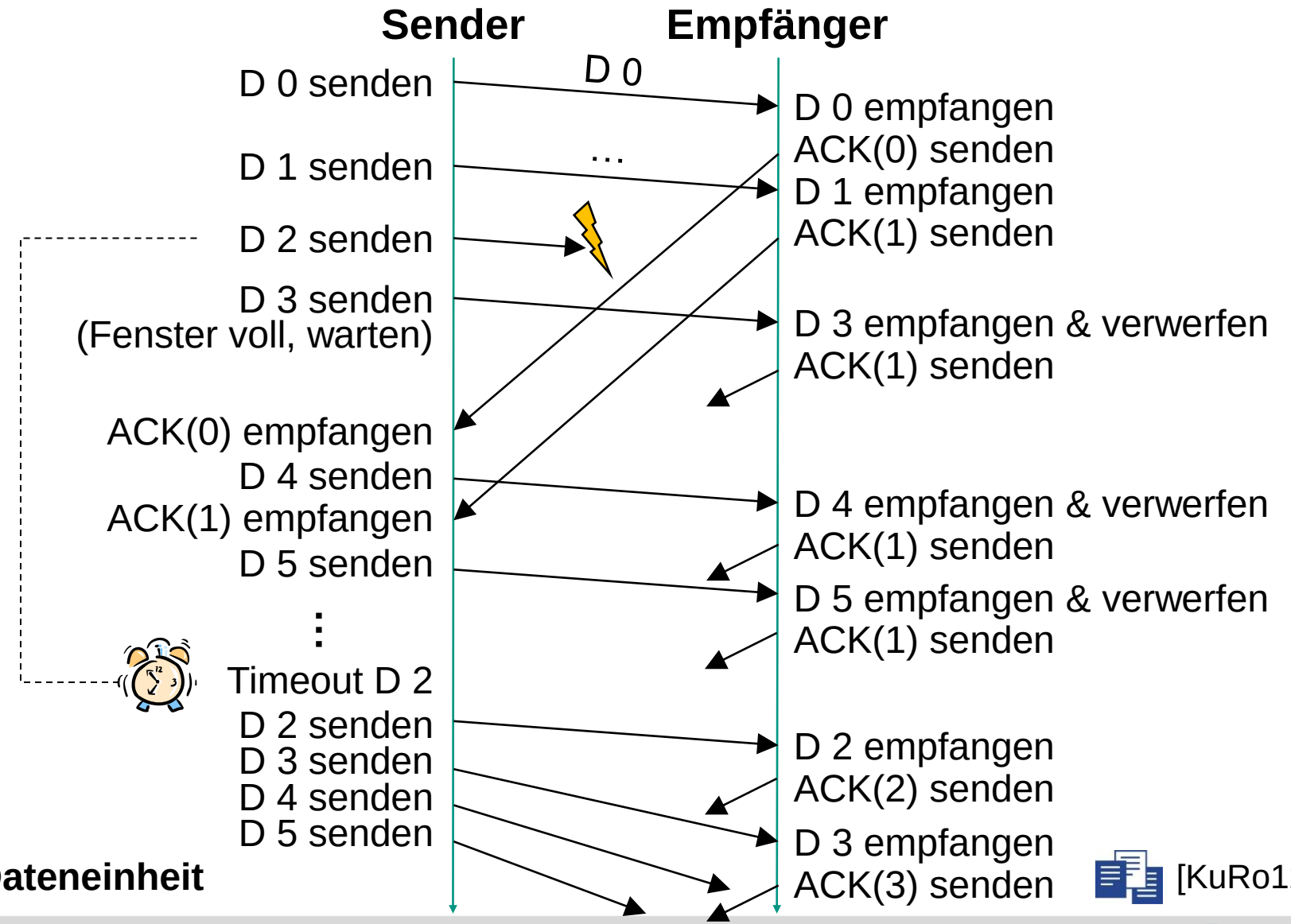
Go-Back-N ARQ

- Ziel
  - Erhöhung der Leistungsfähigkeit im Vergleich zu Stop-and-Wait
- Datenaustausch
  - Sender
    - Kann *mehrere* Dateneinheiten senden bis er Quittung erhalten muss
    - Maximale Anzahl der nicht quittierten Dateneinheiten ist begrenzt
    - Typischerweise durch ein Fenster (Window) auf Senderseite
  - Empfänger
    - Quittiert i.d.R. mit kumulativen Quittungen

Go-Back-N ARQ

- Verhalten im Fehlerfall
  - Empfänger
    - Empfang einer fehlerhaften Dateneinheit oder einer Dateneinheit außerhalb der Reihenfolge
      - Verwerfen aller nachfolgenden Dateneinheiten
  - Sender
    - Ablauf des entsprechenden Zeitgebers
      - Wiederholen aller noch nicht quittierten Dateneinheiten
- Fragen
  - Wo ist Pufferung der Dateneinheiten erforderlich?
  - Wie viele müssen gepuffert werden?

Go-Back-N: Beispielablauf



D = Dateneinheit

[KuRo12]

Go-Back-N: Leistungsbewertung

- Bemerkung
  - Fenstergröße W begrenzt maximale Anzahl gesendeter und noch nicht quittierter Dateneinheiten
- Annahme: Fehlerfreie Übertragung

- Zwei Fälle
  - Fenstergröße $W \geq 1 + 2a$
    - Sender kann ohne Pause senden
    - Übertragungsabschnitt ist 100% ausgelastet
  - Fenstergröße $W < 1 + 2a$
    - Sender kann nach dem Aufbrauchen des Fensters nicht weiter senden

- Erzielbare Auslastung

$$U = \begin{cases} 1 & W \geq 1 + 2a \\ \dfrac{W}{1 + 2a} & W < 1 + 2a \end{cases}$$

Go-Back-N: Leistungsbewertung

- *Jetzt*: Berücksichtigung von Übertragungsfehlern

- Herleitung ähnlich wie bei Stop-and-Wait, aber
  - Im Fehlerfall werden K Dateneinheiten wiederholt anstatt einer
  - $n =$ Erwartungswert [Anzahl übertragener Dateneinheiten für eine erfolgreiche Übertragung]
    - $f(i)$: Anzahl übertragener Dateneinheiten, falls die ursprünglich gesendete Dateneinheit i mal übertragen werden muss

$$f(i) = 1 + (i-1)K = (1-K) + Ki$$

    - Daraus ergibt sich für den Erwartungswert n

$$n = \sum_{i=1}^{\infty} f(i) p^{i-1}(1-p)$$

Go-Back-N: Leistungsbewertung

- Einsetzen von $f(i)$

$$n = \sum_{i=1}^{\infty} ((1-K) + Ki)p^{i-1}(1-p)$$

für $-1 < x < 1$ gilt:

$$\sum_{i=1}^{\infty} x^{i-1} = \frac{1}{1-x}$$

$$n = (1-K)\sum_{i=1}^{\infty} p^{i-1}(1-p) + K\sum_{i=1}^{\infty} ip^{i-1}(1-p)$$

für $-1 < x < 1$ gilt:

$$\sum_{i=1}^{\infty} ix^{i-1} = \frac{1}{(1-x)^2}$$

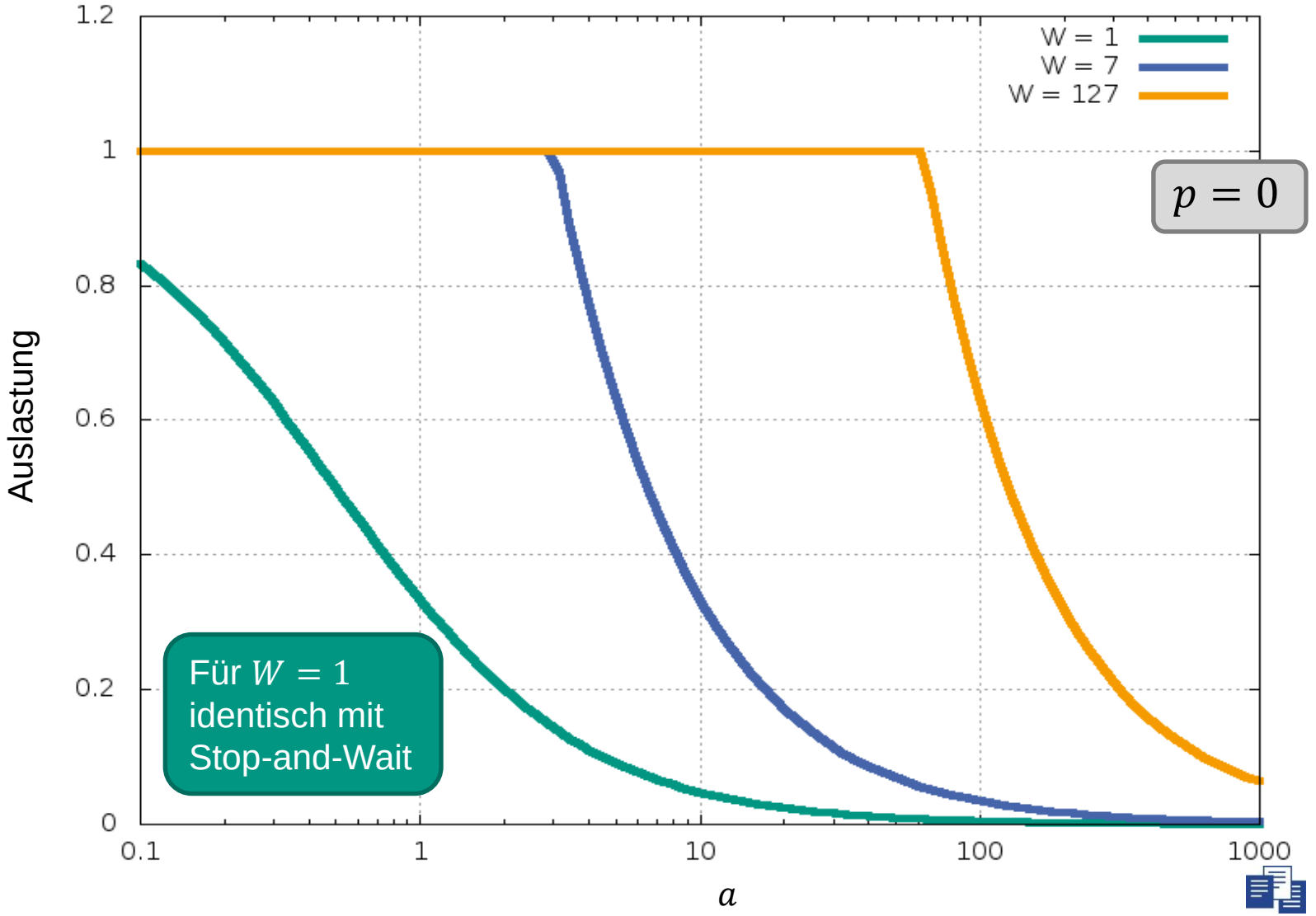
$$n = 1 - K + \frac{K}{1-p} = \frac{1-p+Kp}{1-p}$$

  - Für $W \geq (1+2a)$: K ungefähr $1+2a$
  - Für $W < (1+2a)$: $K = W$

- Damit

$$U = \begin{cases} \dfrac{1-p}{1+2ap} & W \geq 1+2a \\ \dfrac{W(1-p)}{(1+2a)(1-p+Wp)} & W < 1+2a \end{cases}$$

Bewertung von Go-Back-N



[Stal06]

Selective Repeat ARQ

- Ziel
  - Erhöhung der Auslastung im Vergleich zu Stop-and-Wait
  - Reduzierung des Datenaufkommens im Vergleich zu Go-Back-N

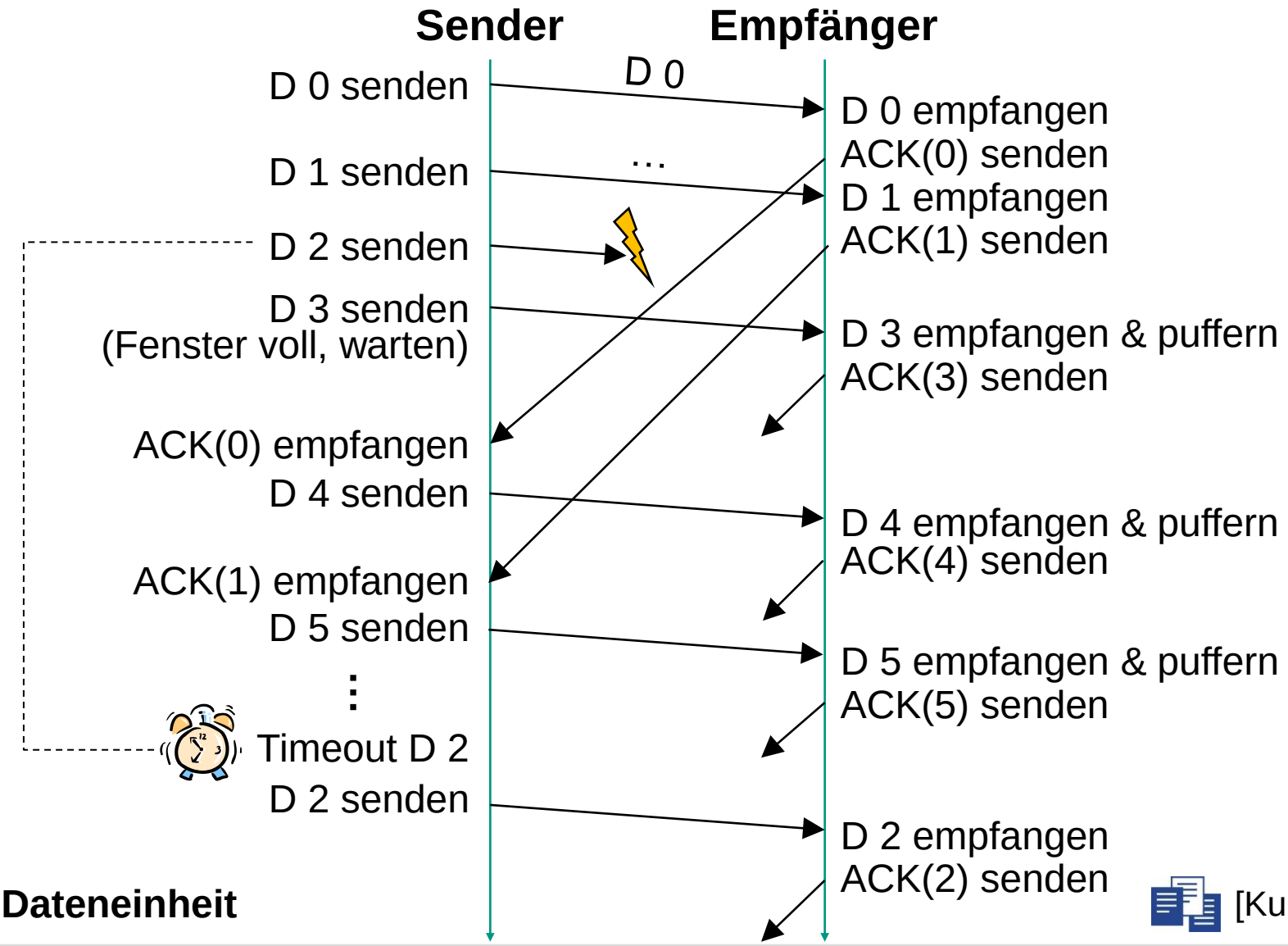
- Datenaustausch
  - Sender wie bei Go-Back-N
    - Sender kann mehrere Dateneinheiten senden, bis er eine Quittung erhalten muss
    - Maximale Anzahl der nicht quittierten Dateneinheiten ist begrenzt
  - Empfänger
    - Quittiert mit selektiven Quittungen

Selective Repeat ARQ

- Verhalten im Fehlerfall
  - Empfänger
    - Nachfolgende, korrekt empfangene Dateneinheiten werden vom Empfänger gepuffert und bestätigt
  - Sender
    - Nur nicht korrekt empfangene Dateneinheiten werden vom Sender wiederholt

- Fragen
  - Wo ist eine Pufferung der Dateneinheiten erforderlich? Wie viele müssen gepuffert werden?
  - Vor- und Nachteile von Go-Back-N und Selective Repeat im Vergleich?

Selective Repeat: Beispielablauf



D = Dateneinheit

[KuRo12]

Varianten in den ARQ-Verfahren

- Negative Quittungen (NACKs)
  - Nicht korrekt empfangene Dateneinheiten werden mit negativer Quittung (NACK) bestätigt
  - Go-Back-N
    - Sender wiederholt ab dieser Sequenznummer alle gesendeten Dateneinheiten
  - Selective Reject
    - Sender wiederholt genau die Dateneinheit mit dieser Sequenznummer
- Kumulative Quittungen (Go-Back-N)
  - Quittung erfolgt für mehrere Dateneinheiten auf einmal
  - Kumulative Sequenznummer gibt an, bis wohin die Daten korrekt empfangen wurden, d.h. es handelt sich um positive Quittungen

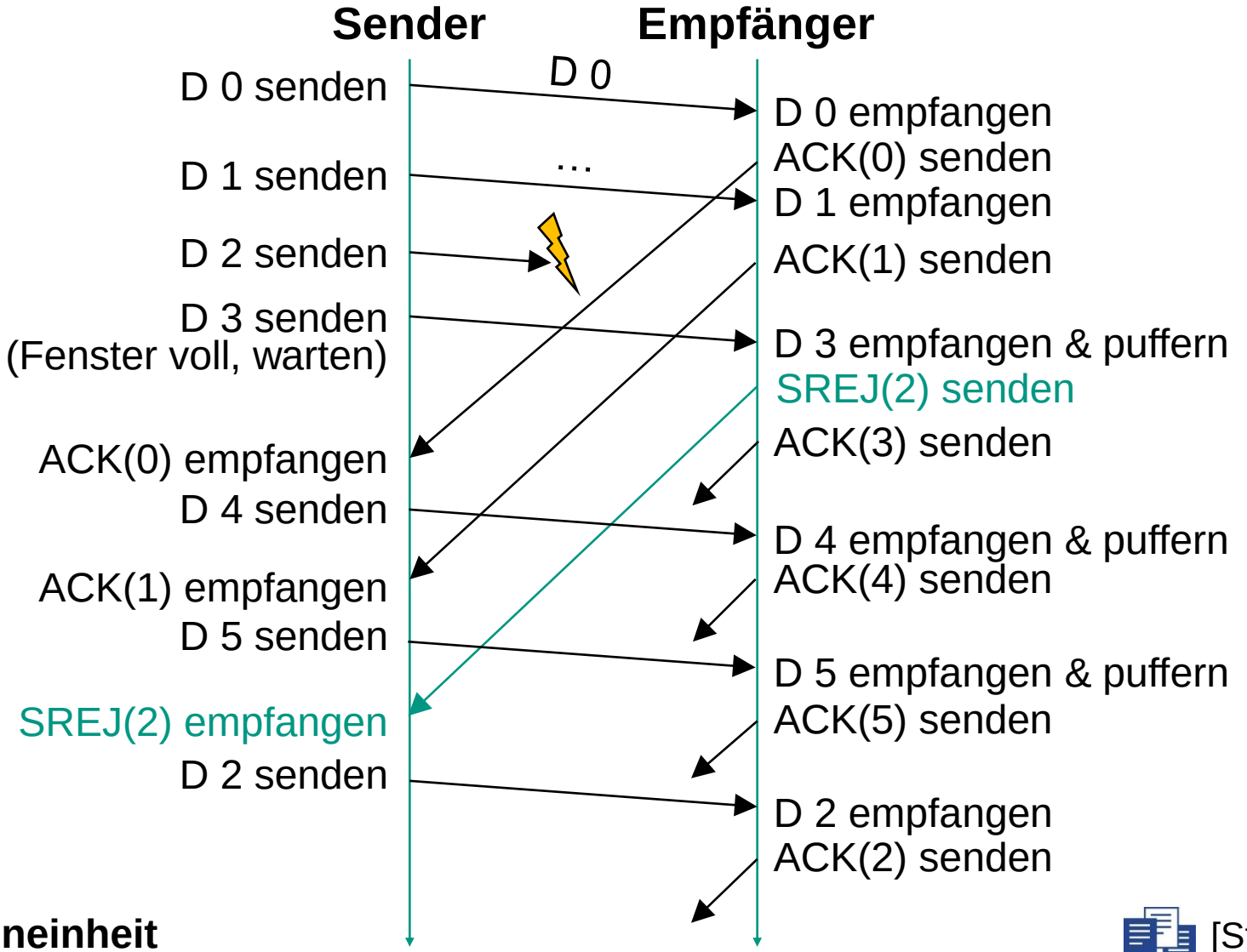
Selective Repeat vs. Selective Reject

- Bis auf das Quittierungsverhalten des Empfängers identisch

- Selective Repeat
  - Fehlerhafte Dateneinheit wird nicht vom Empfänger bestätigt
  - Sender wiederholt diese Dateneinheit nach Timeout

- Selective Reject
  - Empfänger teilt dem Sender mit einer negativen Quittung SREJ(Sequenz-Nr.) mit, dass Dateneinheit nicht korrekt empfangen wurde
  - Sender wiederholt sofort die Dateneinheit und wartet nicht auf Timeout

Selective Reject: Beispielablauf



D = Dateneinheit

[Stal10]

Selective Reject: Leistungsbewertung

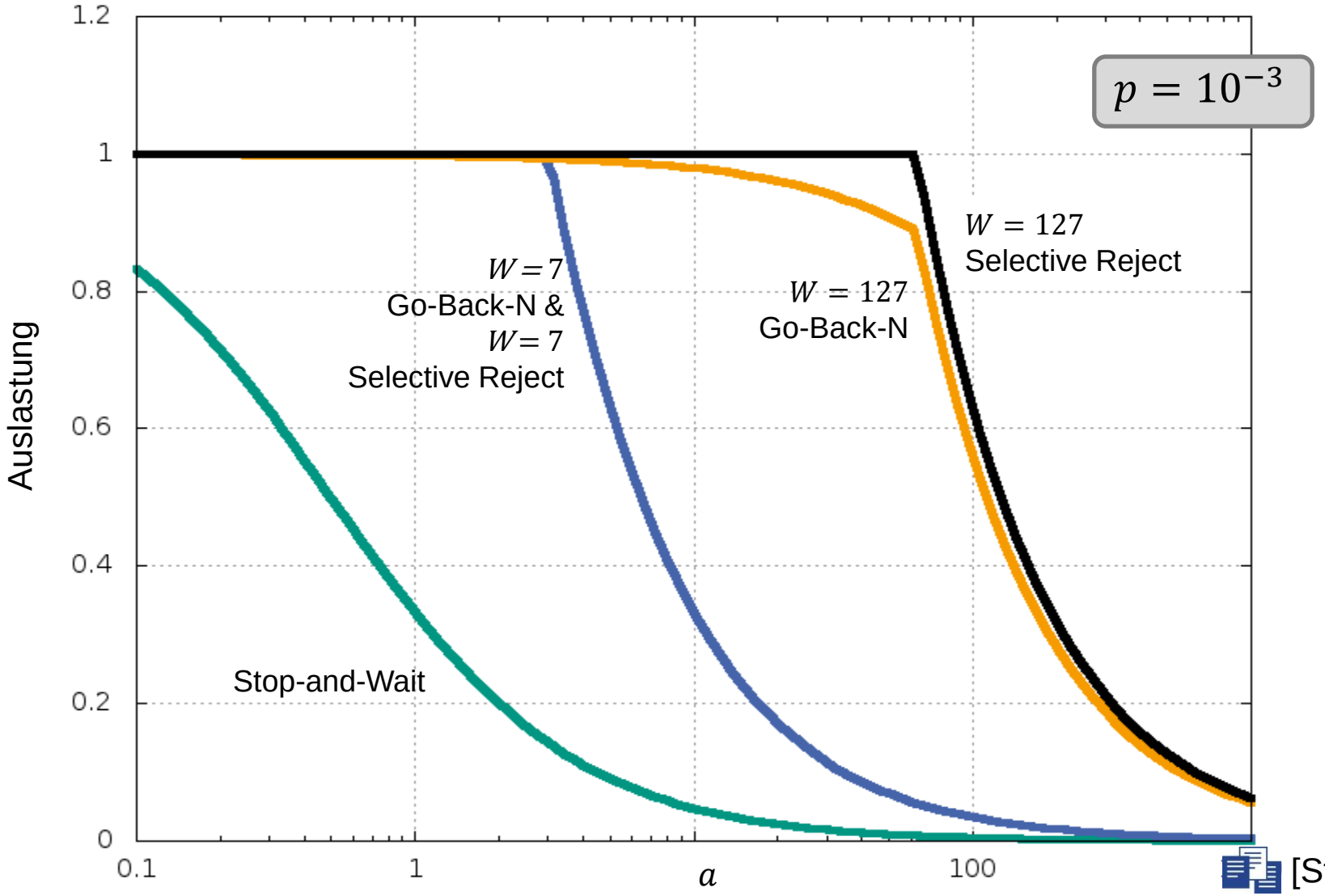
- Hier: mit Übertragungsfehlern
- Herleitung analog zu Stop-and-Wait
  - Es wird immer nur eine Dateneinheit wiederholt

$$n = \frac{1}{1-p}$$

  - Fallunterscheidung wegen Fenster

$$U = \begin{cases} 1-p & W \geq 1+2a \\ \dfrac{W(1-p)}{1+2a} & W < 1+2a \end{cases}$$

Vergleichende Bewertung



[Stal06]

4.5.5 Vorwärtsfehlerkorrektur

- Vorwärtsfehlerkorrektur (*Forward Error Correction* – FEC)
  - Fehlerkorrigierende Codes
  - Soll dazu dienen, verloren gegangene Dateneinheiten zu rekonstruieren

- Beispiel
  - Zu senden sind die Dateneinheiten

0101 – D1
1111 – D2
0000 – D3

  - Dazu wird über XOR eine weitere Dateneinheit berechnet

1010 – D4

  - Diese vier Dateneinheiten werden an Empfänger gesendet

[Hals05]

Vorwärtsfehlerkorrektur – Ablauf

- Empfänger muss nur drei der vier Dateneinheiten korrekt empfangen, um fehlende Dateneinheit rekonstruieren zu können
  - Er verknüpft die korrekt empfangenen Dateneinheiten mit XOR und rekonstruiert so die fehlende:
    - D1 geht verloren

      1111
      0000
      1010

      0101 – D1
    - D2 geht verloren

      0101
      0000
      1010

      1111 – D2
    - D3 geht verloren

      0101
      1111
      1010

      0000 – D3
  - Der Empfänger muss wissen, welche Dateneinheit verloren ging

Kapitelübersicht

1. Einführung
2. Netzwerkarchitekturen
3. Physikalische Grundlagen
4. Protokollmechanismen
5. Die Sicherungsschicht: HDLC
6. Die Sicherungsschicht: Lokale Netze
7. Netzkopplung und Vermittlung
8. Die Transportschicht
9. Sicherheit
10. Anwendungssysteme

1. Basis-Szenario
2. Fehlertypen und Fehlerursachen
3. Mechanismen zur Fehler-erkennung und -behebung
4. Fehlerkontrolle bei Bitfehlern
5. Fehlerkontrolle bei Paketfehlern
6. Flusskontrolle
7. Verbindungen
8. Zusammenfassung

4.6 Flusskontrolle

- Problem
  - Empfänger kann von Sender überlastet werden
    - Daten können nicht empfangen werden, da Puffer nicht ausreichend
    - → Datenverluste
  - Sender muss Größe des Empfangspuffers berücksichtigen

- Anforderungen
  - Einfachheit
  - Möglichst geringe Nutzung von Netzressourcen
  - Fairness
  - Stabilität

Flusskontrolle

- Varianten
  - Closed Loop
    - Rückkopplung, um zu verhindern, dass Empfänger „überschwemmt“ wird
    - Sender adaptiert ihren Datenstrom entsprechend
  - Open Loop
    - Beschreibung des Verkehrs mit anschließender Ressourcenreservierung und Überwachung des eingehenden Verkehrs

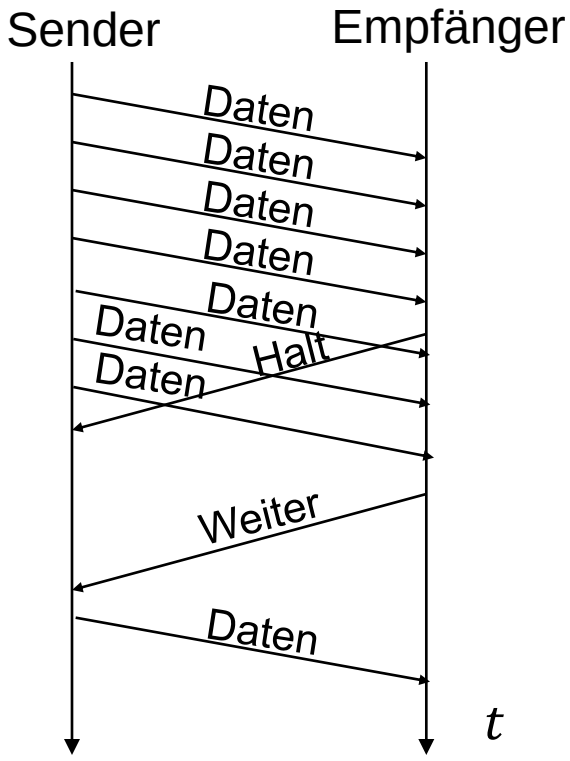
    … hier nicht weiter behandelt

Closed-Loop Flusskontrolle

- Varianten
  - Halt-und-Weiter
  - Stop-and-Wait
    - Kombiniert Fehler- und Flusskontrolle
  - Kreditbasierte Flusskontrolle
    - Statisches Fenster
    - Dynamisches Fenster

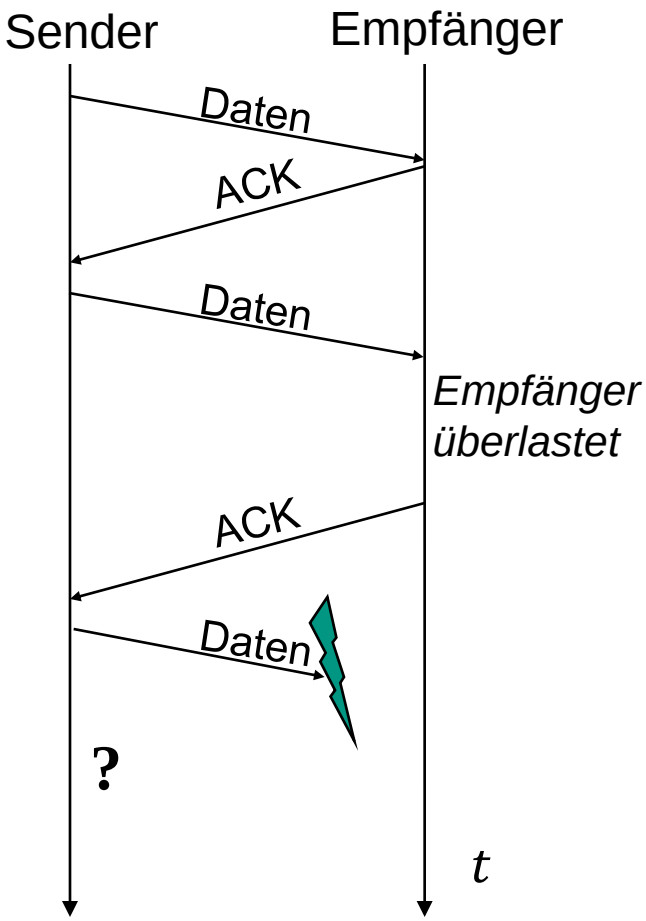
Halt-und-Weiter

- Sehr einfache Methode
  - Meldungen
    - Halt
    - Weiter
  - Kann Empfänger nicht mehr Schritt halten, schickt er eine Halt-Meldung
  - Ist Empfang wieder möglich, sendet Empfänger eine Weiter-Meldung
- Bewertung
  - Nur auf Vollduplex-Leitungen verwendbar
  - Bei hohen Verzögerungen nicht effektiv
  - Probleme bei Verlust der Halt-Meldung
- Beispiel
  - Fast-Ethernet, Gigait-Ethernet



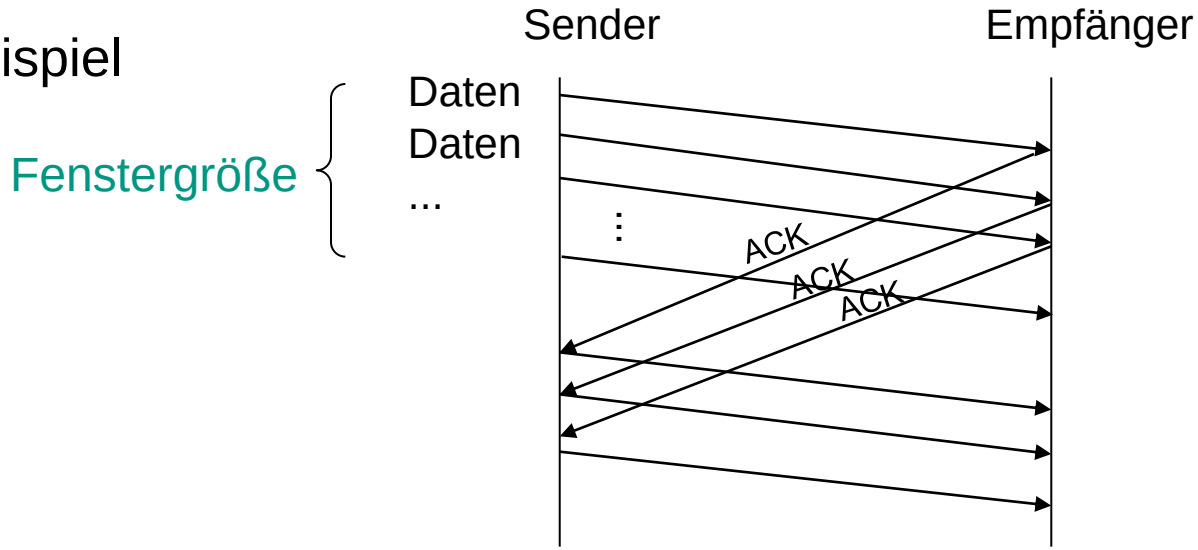
Stop-and-Wait

- Funktionsweise
  - Durch Zurückhalten der Quittung kann der Sender gebremst werden
  - Verfahren zur Fehlererkennung wird hier
  - für Flusskontrolle mitbenutzt wird
- Problem
  - Der Sender kann nicht mehr unterscheiden,
    - ob Dateneinheit verloren ging, oder
    - ob Empfänger die Quittung wegen Überlast zurückgehalten hat



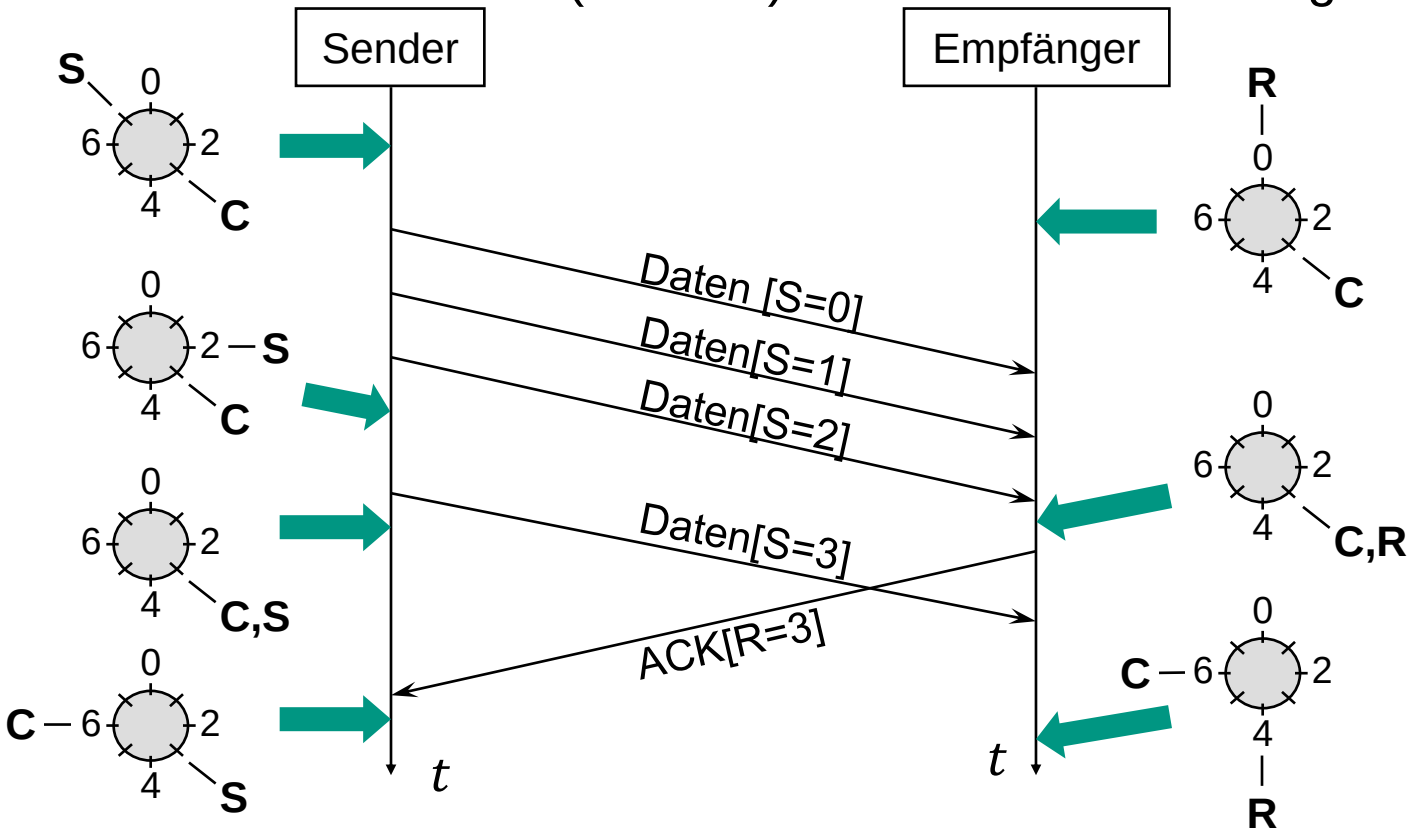
Kreditbasierte Flusskontrolle

- Prinzip
  - Sender kann bis zu einer maximalen Anzahl Dateneinheiten (bzw. Bytes) senden, ohne eine Quittung zu empfangen
  - Maximale Anzahl der Dateneinheiten repräsentiert die Pufferkapazität des Empfängers und wird als (Sende-)Kredit bezeichnet
  - Oftmals als fortlaufendes Fenster bezeichnet (engl.: Sliding Window)
    - Fenster wird mit jeder empfangenen reihenfolgetreuen positiven Quittung weitergeschaltet
    - Empfänger kann meist zusätzlich den Kredit explizit bestimmen (z.B. in TCP)
- Beispiel



Kreditbasierte Flusskontrolle: Sliding Window

Beispiel: Fenstermechanismus (Kredit 4) für eine Senderichtung

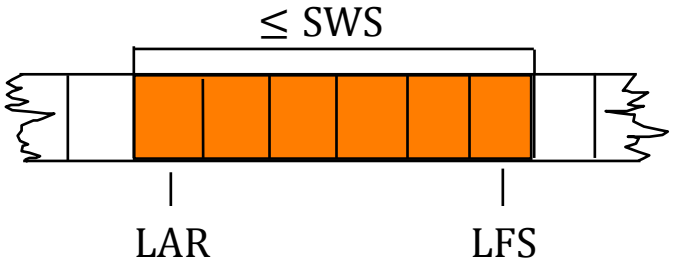


S: Sende-Sequenznummer (der zuletzt gesendeten Dateneinheit)
R: Nächste erwartete Sende-Sequenznummer = Quittierung bis Empfangs-Sequenznummer R – 1
C: Oberer Fensterrand (maximal erlaubte Sequenznummer)

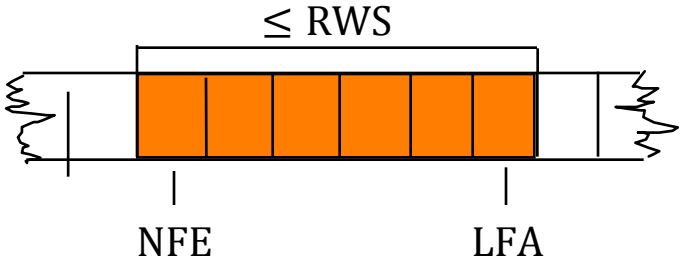
Nachteil: Kopplung von Fluss- und Fehlerkontrolle

Flusskontrolle mit Sliding Window

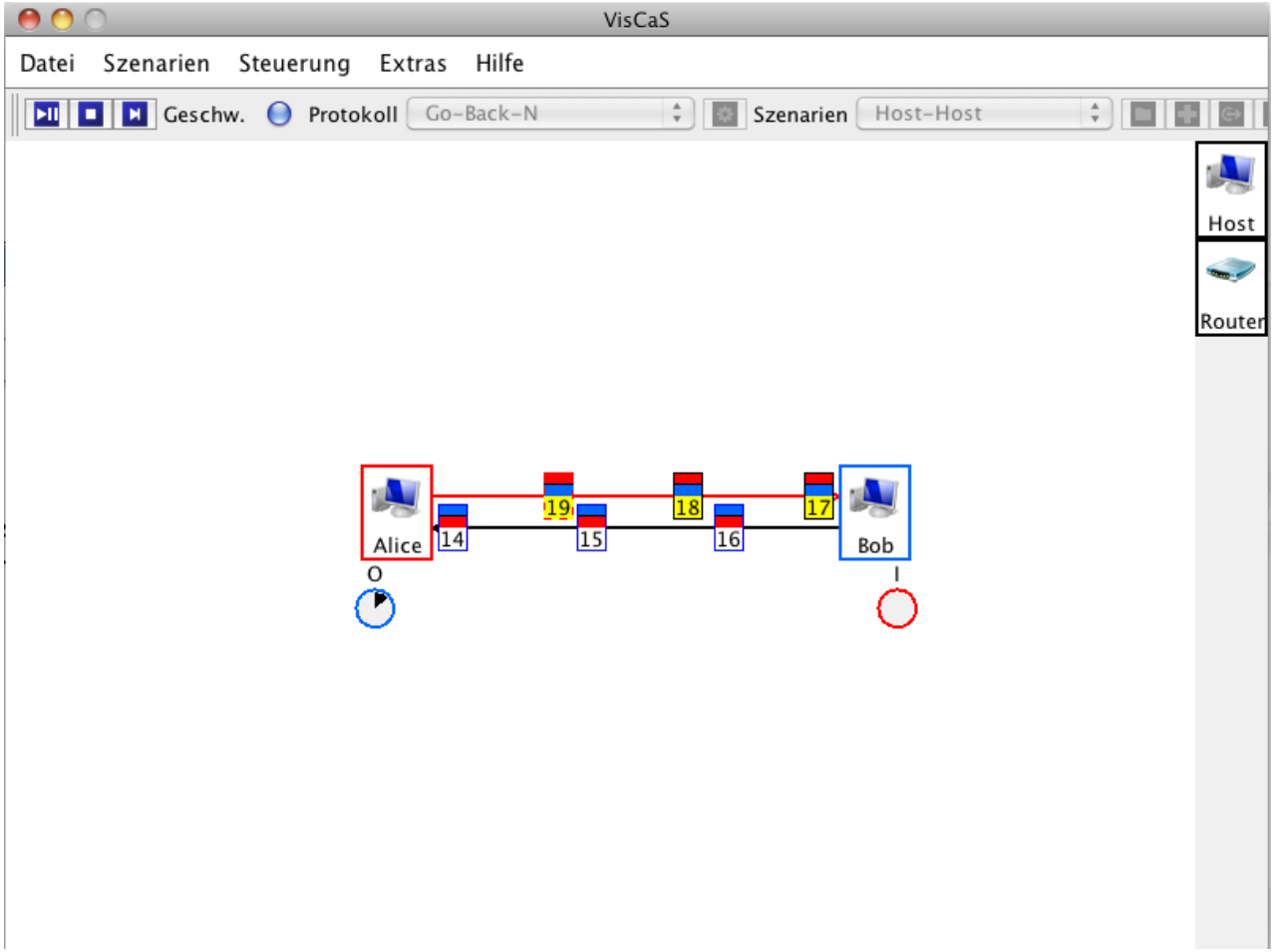
- Sender
  - SWS: Send Window Size (max. Anzahl ausstehender Dateneinheiten bzw. Bytes)
  - LAR: Last ACK Received (Sequenznummer der nächsten erwarteten Dateneinheit bzw. Bytes)
  - LFS: Last Frame Sent (Sequenznummer der letzten gesendeten Dateneinheit bzw. Bytes)
- Invariante
  - $\text{LFS} - \text{LAR} + 1 \leq \text{SWS}$



- Empfänger
  - RWS: Receiver Window Size (max. Anzahl nicht in Reihenfolge empfangener Dateneinheiten bzw. Bytes)
  - LFA: Last Frame Acceptable (Sequenznummer der letzten empfangbaren Dateneinheit bzw. Bytes)
  - NFE: Next Frame Expected (Sequenznummer der nächsten in Reihenfolge erwarteten Dateneinheit bzw. Bytes)
- Invariante
  - $\text{LFA} - \text{NFE} + 1 \leq \text{RWS}$

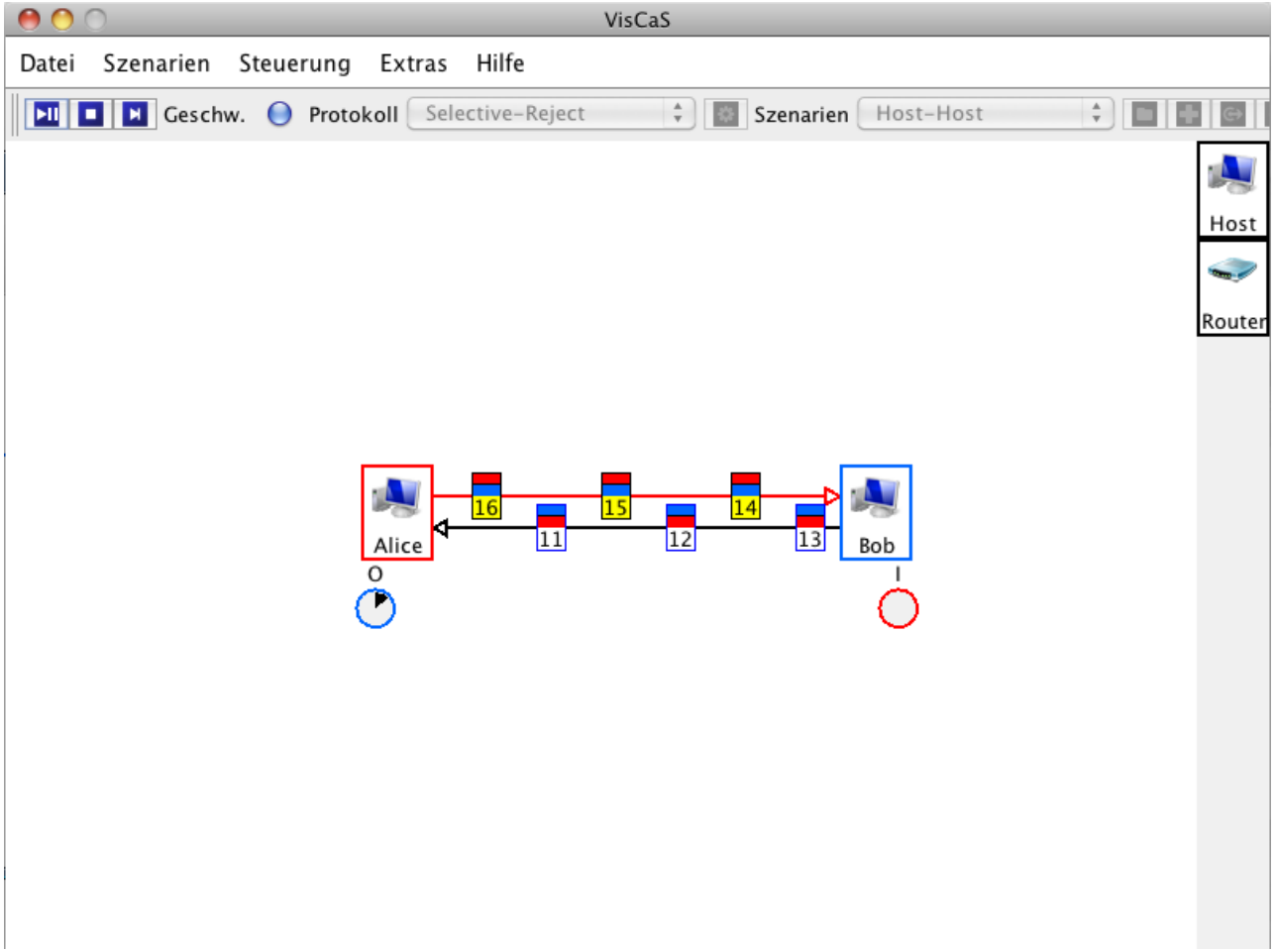


Visualisierung: Go-back-N und Flusskontrolle



http://www.tm.kit.edu/software/viscas/

Visualisierung: Selective Reject und Flusskontrolle



http://www.tm.kit.edu/software/viscas/

Kapitelübersicht

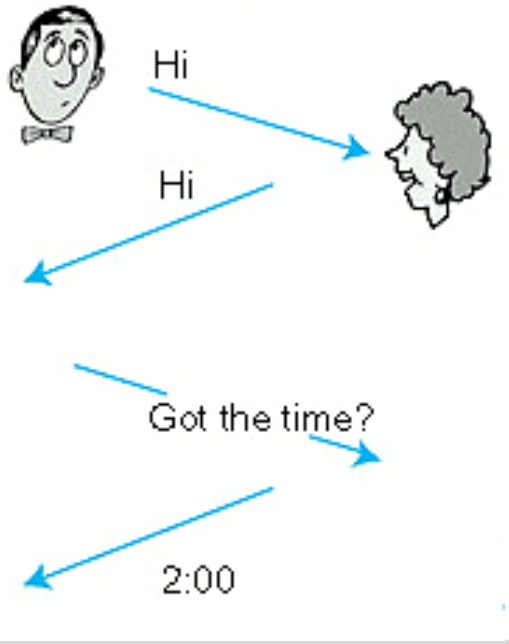
1. Einführung
2. Netzwerkarchitekturen
3. Physikalische Grundlagen
4. Protokollmechanismen
5. Die Sicherungsschicht: HDLC
6. Die Sicherungsschicht: Lokale Netze
7. Netzkopplung und Vermittlung
8. Die Transportschicht
9. Sicherheit
10. Anwendungssysteme

1. Basis-Szenario
2. Fehlertypen und Fehlerursachen
3. Mechanismen zur Fehler-erkennung und -behebung
4. Fehlerkontrolle bei Bitfehlern
5. Fehlerkontrolle bei Paketfehlern
6. Flusskontrolle
7. Verbindungen
8. Zusammenfassung

4.7 Verbindungen

- Verbindungen
  - Stellen Kommunikationsbeziehung zwischen Kommunikationspartnern her
  - Bei jedem Kommunikationspartner wird ein Verbindungskontext etabliert
    - Sequenznummern, Fenstergröße etc.
    - Basis für die Bereitstellung zuverlässiger Kommunikationsdienste

- Beispiel



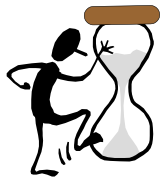
Verbindungslos vs. Verbindungsorientiert

- Verbindungslose Kommunikation
  - Daten werden versendet, ohne vorherigen Aufbau einer Verbindung
    - Im Beispiel würde also die Begrüßung wegfallen
  - Vorteil
    - schnelle Datenversendung möglich
  - Nachteil
    - keine Möglichkeit der Kontrolle, ob der Kommunikationspartner überhaupt zuhört bzw. zuhören kann

Verbindungslos vs. Verbindungsorientiert

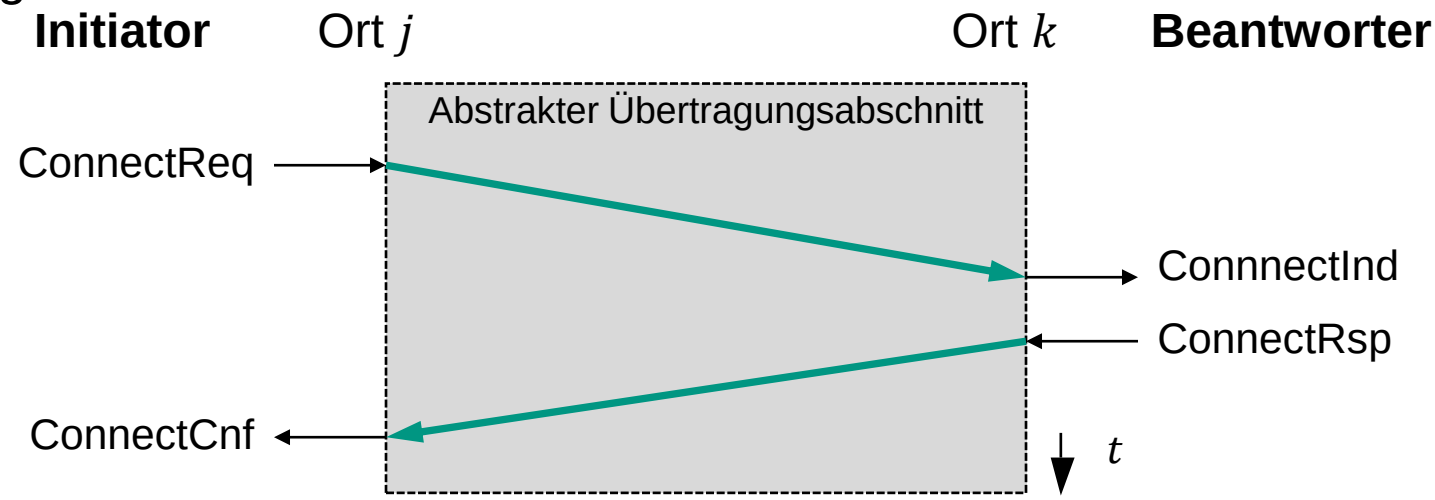
- Verbindungsorientierte Kommunikation
  - Dem Datenaustausch geht *Aufbau* einer Verbindung voraus
  - Am Ende erfolgt der *Abbau* einer Verbindung
  - Vorteil
    - Aushandlung von Kommunikationsparametern möglich
      - Z.B. benötigte Fenstergrößen, verwendete Fehlerkontrollmechanismen, Sequenznummern ...
  - Nachteile
    - Eigentlicher Datenaustausch verzögert
    - Overhead der Verbindungsetablierung und -verwaltung kann höher sein als der eigentliche Datenaustausch

    → häufig beim Aufruf von Webseiten der Fall

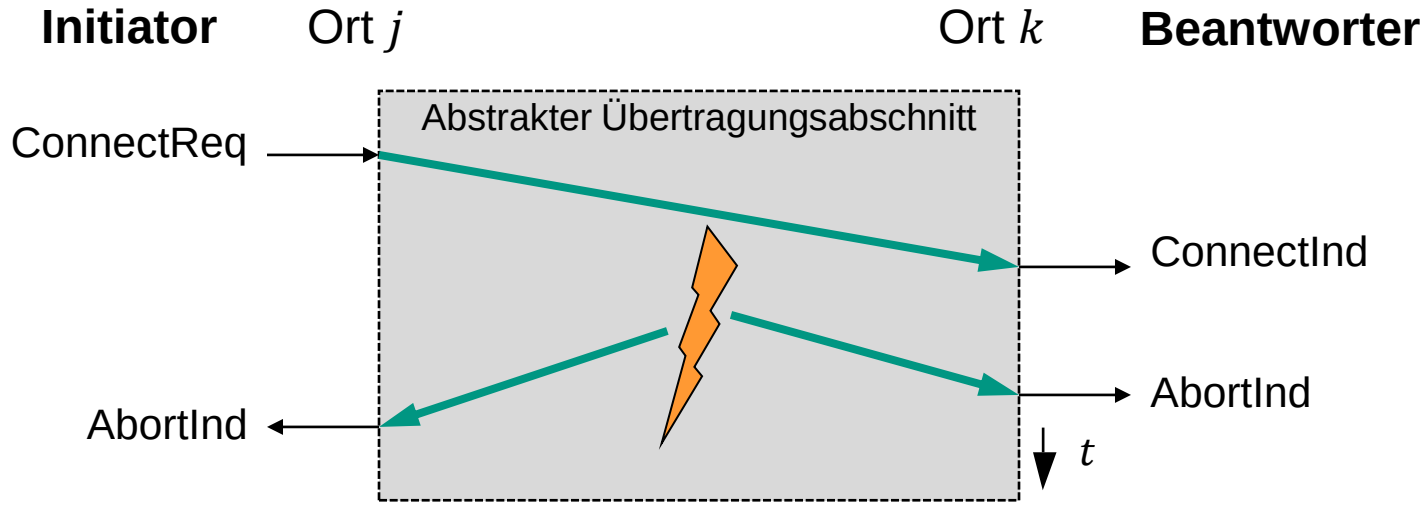


Verbindungsaufbau: 2-Wege-Handshake

- Erfolgreicher Aufbau

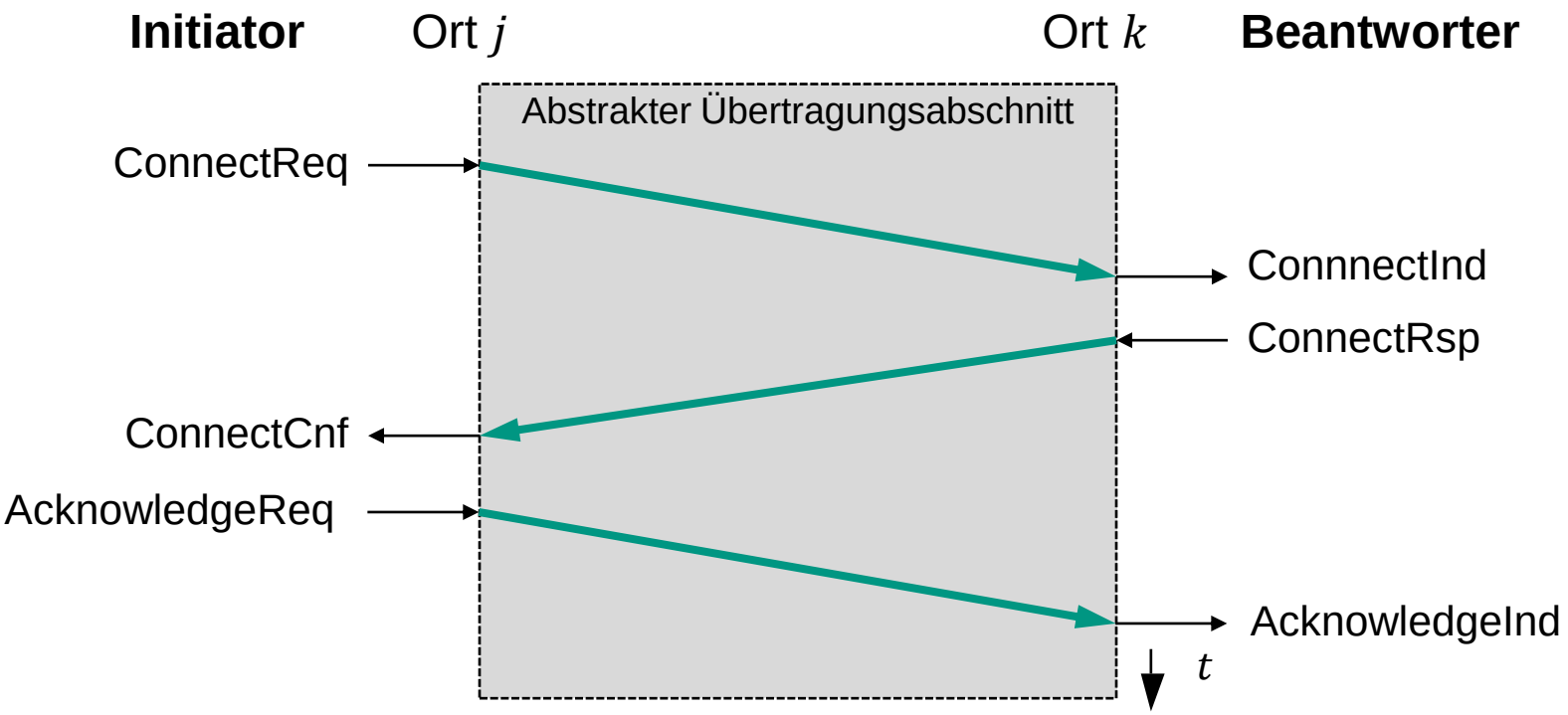


- Beispiel nicht erfolgreicher Aufbau



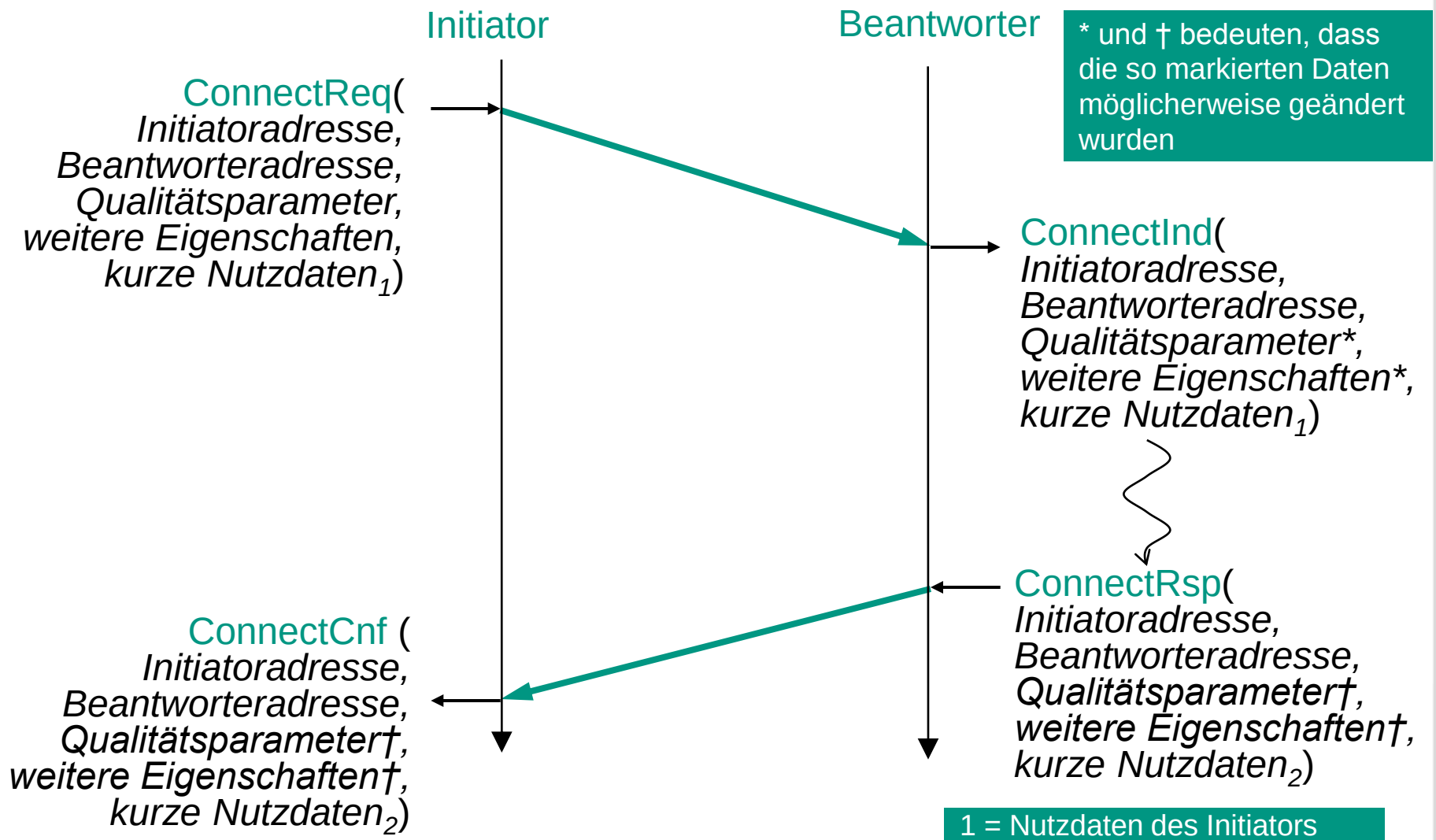
Verbindungsaufbau: 3-Wege-Handshake

- Erfolgreicher Aufbau



- … wird beispielsweise beim Protokoll TCP verwendet

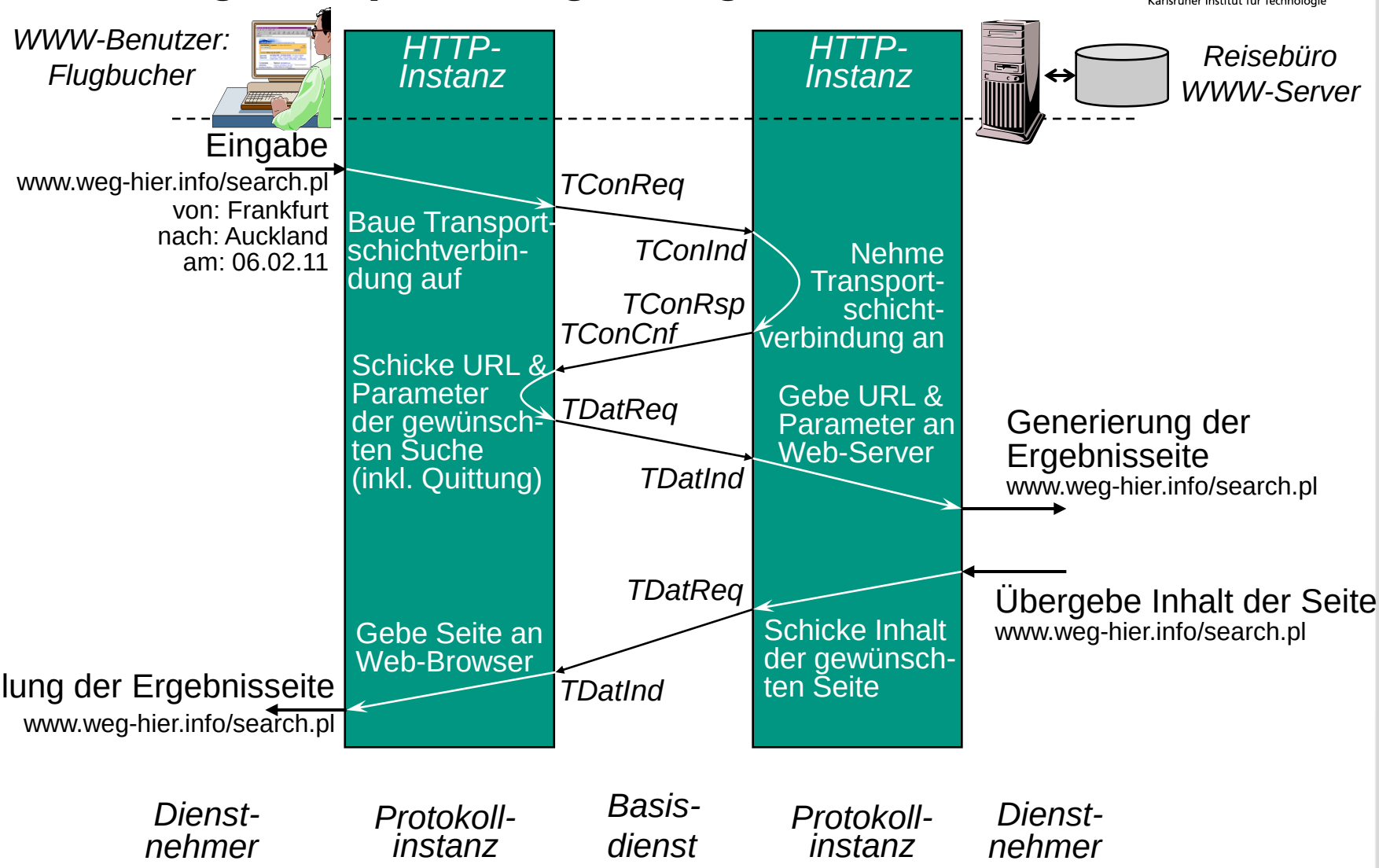
Beispiel: Verbindungsaufbau



* und † bedeuten, dass die so markierten Daten möglicherweise geändert wurden

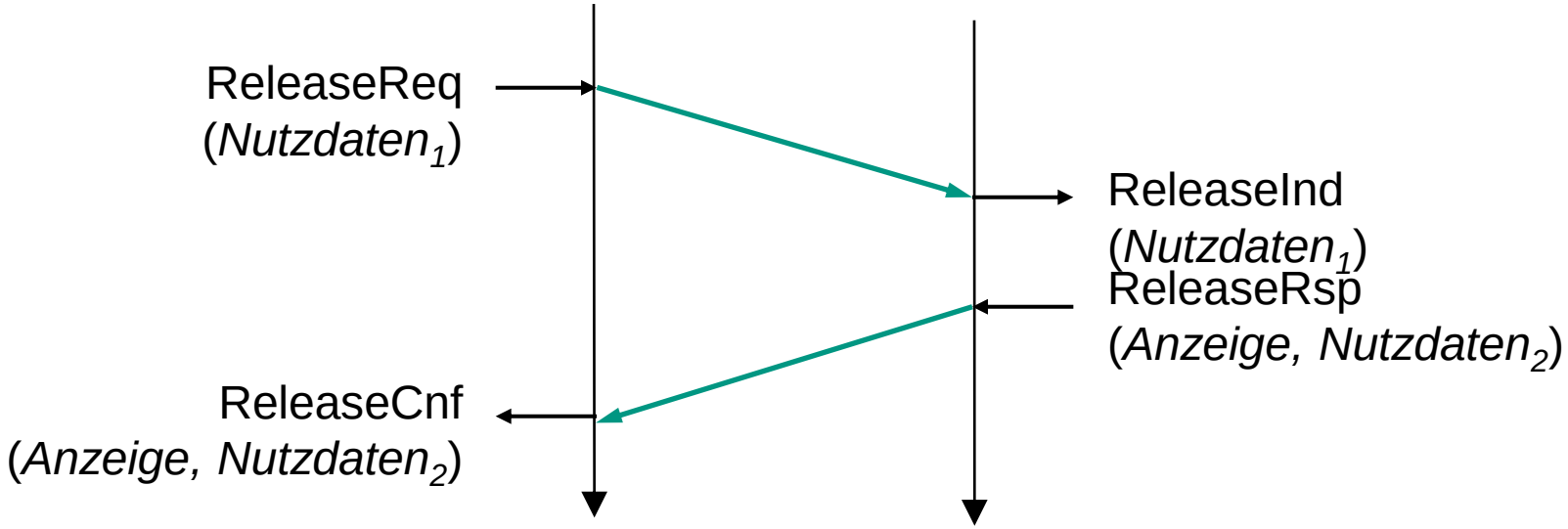
1 = Nutzdaten des Initiators
2 = Nutzdaten des Beantworters

Anwendungsbeispiel: Fluganfrage im Internet



Beispiel: Verbindungsabbau

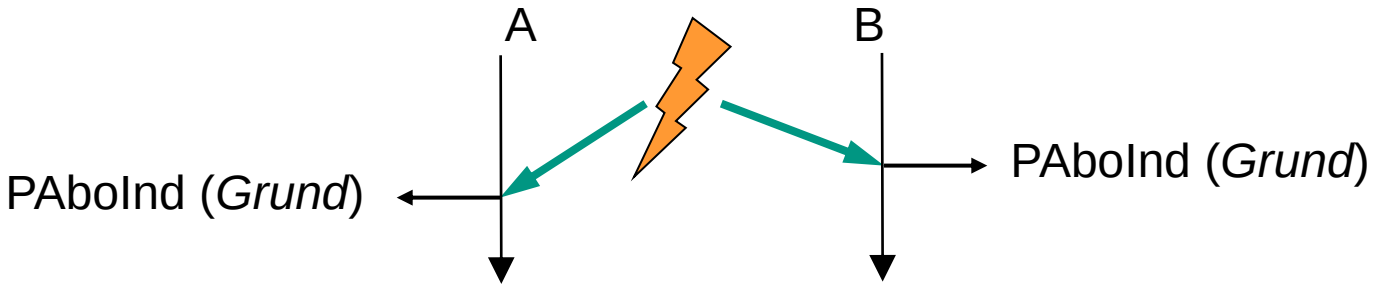
- Geregelter Verbindungsabbau
  - Partner kann Verbindungsabbau zustimmen oder ihn ablehnen



1 = Nutzdaten des Initiators
2 = Nutzdaten des Beantworters

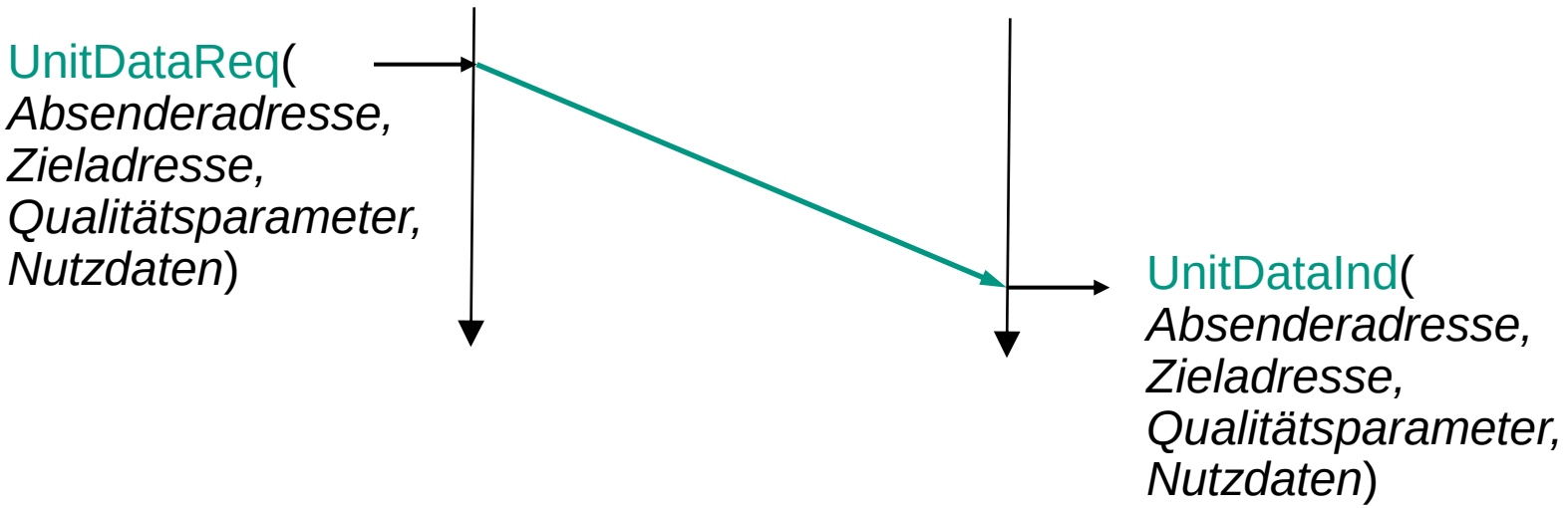
Beispiel: Verbindungsabbruch

- Außerplanmäßiger Abbruch einer Verbindung
  - Erbringerabbruch (Provider Abort, PAbo)
    - Kann in jedem Zustand einer Verbindung passieren
    - Keine Gleichzeitigkeit / bestimmte Reihenfolge garantiert



  - Nutzerabbruch (User Abort, UAbo)
    - Verbindung gilt für Initiator sofort als abgebrochen

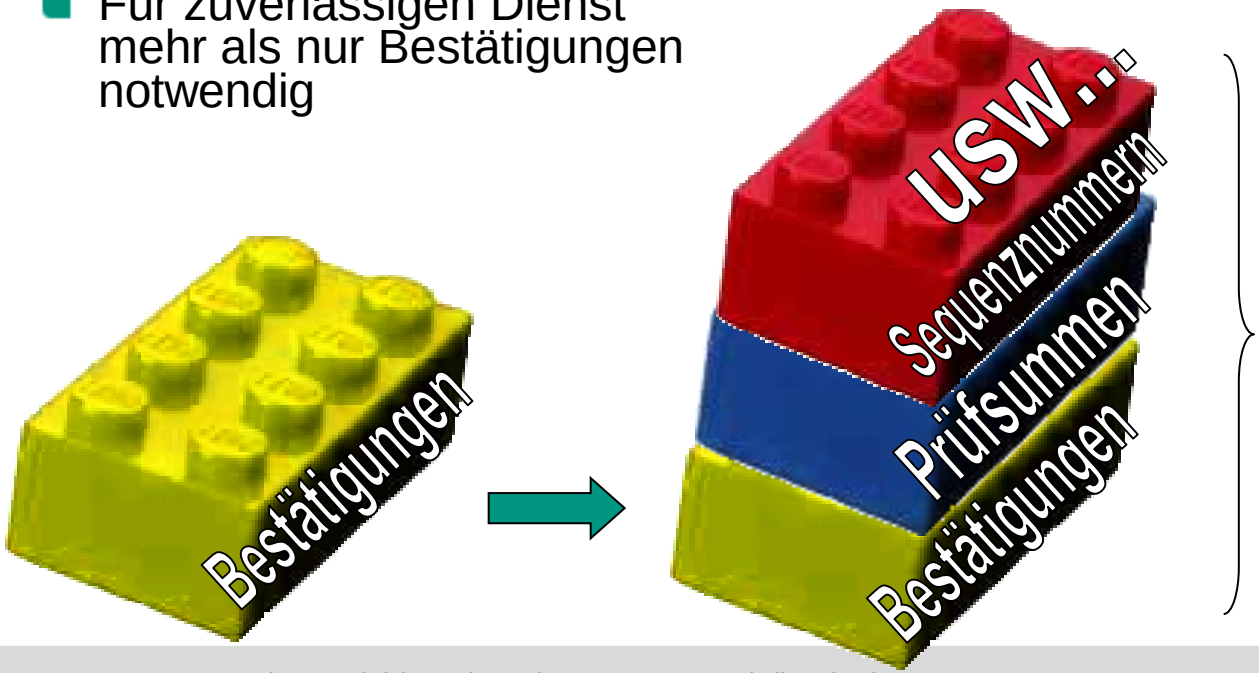
Beispiel: Datagramm-Dienste



- Keine Verbindungen zwischen Kommunikationspartnern
  - Kein Zusammenhang zwischen verschiedenen Übertragungsleistungen
  - Unterstützt keine Auslieferungsdisziplin
    - z.B. keine Garantie für Reihefolgetreue
- Datagramm-Dienst realisiert unzuverlässige Dienstleistung
  - keine Aushandlung zwischen Kommunikationspartnern

Anmerkung: Bestätigt vs. Zuverlässig

- Bestätigt ≠ Zuverlässig
  - Zuverlässiger Dienst stellt sicher, dass alle Daten korrekt übertragen wurden
    - Alle Daten korrekt und vollständig
    - In der richtigen Reihenfolge
    - Ohne Duplikate
    - Ohne Phantom-Dateneinheiten
  - Bestätigungen nur ein Baustein für zuverlässige Dienste
    - Für zuverlässigen Dienst mehr als nur Bestätigungen notwendig



Auszug aus möglichem Aufbau eines zuverlässigen Dienstes

Kapitelübersicht

1. Einführung
2. Netzwerkarchitekturen
3. Physikalische Grundlagen
4. Protokollmechanismen
5. Die Sicherungsschicht: HDLC
6. Die Sicherungsschicht: Lokale Netze
7. Netzkopplung und Vermittlung
8. Die Transportschicht
9. Sicherheit
10. Anwendungssysteme

1. Basis-Szenario
2. Fehlertypen und Fehlerursachen
3. Mechanismen zur Fehler-erkennung und -behebung
4. Fehlerkontrolle bei Bitfehlern
5. Fehlerkontrolle bei Paketfehlern
6. Flusskontrolle
7. Verbindungen
8. Zusammenfassung

4.8 Zusammenfassung

- Bei der Übertragung von Daten können Fehler auftreten
  - Unterschiedliche Fehlertypen und -ursachen
- Grundlegende Protokollmechanismen
  - Fehlerkontrolle (bei Bit- und Paketfehlern)
  - Flusskontrolle
  - Verbindungsmanagement

  →Werden uns als Bausteine in vielen Protokollen immer wieder begegnen
- Grundlegende Techniken der Leistungsbewertung
  - Vereinfachungen, aber grundlegende Aussagen möglich

Zusammenfassung

Fehlerkontrolle bei Bitfehlern

Prüfsummen
Vorwärtsfehler-korrektur

Fehlerkontrolle bei Paketfehlern

ARQ-Verfahren:

Stop-and-Wait
Go-Back-N
Selective Repeat
Selective Reject

Mechanismen:

Sequenznummern
Zeitgeber
Quittungen
Sendewiederholungen

Flusskontrolle

Halt/Weiter
Sliding Window

Verbindungs-management

3-Wege-Handshake

Übungen

4.1 Es sind Daten im Umfang von $1\ \mathrm{Mbyte}$ zu übertragen. Die Größe einer Dateneinheit betrage $2000\ \mathrm{Byte}$, die Ausbreitungsverzögerung betrage $20\ \mathrm{ms}$. Vergleichen Sie die erzielbare Auslastung von Stop-and-Wait mit Go-Back-N.

4.2 Wann versendet der Empfänger jeweils Quittungen?

4.3 Welches Ziel hat die Flusskontrolle?

4.4 Konstruieren Sie ein Schieberegister, dass Sie für die Division durch $x^7 + x^5 + x^4 + x + 1$ verwenden können

4.5 Berechnen Sie den CRC für die Dateneinheit 11011000110101 mit dem Generatorpolynom 110011

4.6 Knoten A sendet mit Go-Back-N und Sliding-Window Daten an Knoten B. Die Fenstergröße betrage $4\ \mathrm{Bit}$. Zeichnen sie die Fenster auf beiden Seiten

- Bevor A anfängt Daten zu senden
- Nachdem A die Dateneinheiten 0, 1 und 2 gesendet und B 0 und 1 quittiert hat
- Nachdem A die Dateneinheiten 3, 4 und 5 gesendet hat und B 4 quittiert hat

4.7 Welche Fehlerarten kennen Sie?

4.8 Eine Störung von $10\ \mathrm{ms}$ führt bei einer Datenrate von $100\ \mathrm{Mbit/s}$ zu wie vielen gestörten Bits?

Literatur

[Benv05] Ch. Benvenuti; Understanding Linux Network Internals; O'Reilly; 2005

[Hals05] F. Halsall; Computer Networking and the Internet; Addison-Wesley; 2005
- Kapitel 1.4, Anhang C

[Holz91] G. J. Holzmann; Design and Validation of Computer Protocols; Prentice Hall; 1991

[KuRo12] James Kurose, Keith Ross, Computer Networking, 6/e, Pearson; 2012

[Stal06] W. Stallings; High-Speed Networks: TCP/IP and ATM Design Principles, Prentice Hall, 2006
- Kapitel 9

[Stal10] W. Stallings; Data and Computer Communications, Prentice Hall, 2010
- Kapitel 7