

# Datenkommunikation

Teil 2.2: Sicherungsschicht

O.Univ.Prof.Dr. Harmen R. van As

## Übersicht

### 2.2 OSI-Referenzmodell: Schicht 2 - Sicherung

- Aufgaben und Funktionen
- Dienstelemente
- Fehlererkennung und Fehlererbehandlung
- Flusskontrolle

## Dienste der Sicherungsschicht (1)

### (1) Betrieb von Sicherungsverbindungen

zwischen zwei Instanzen können mehrere Sicherungsverbindungen betrieben werden, Auf- und Abbau dieser Verbindungen geschieht dynamisch.

### (2) Übertragung von Sicherungs-Dienstdateneinheiten (DL-SDU)

maximal erlaubte Länge der LD-SDUs kann begrenzt sein, abhängig von Fehlercharakteristik der physikalischen Verbindung und Fähigkeit der Sicherungsschicht, Übertragungsfehler zu erkennen und zu korrigieren

### (3) Einrichtung von Verbindungsendpunkt-Identifikatoren

jeder Verbindung werden lokal eindeutige Verbindungsendpunkt-Identifikatoren zugeordnet, um eintreffende Daten einer dieser Verbindungen zuzuordnen.

### (4) Reihenfolgeerhaltung (Sequencing)

Weiterleitung aller Pakete in der korrekten Reihenfolge

### (5) Benachrichtigung über Fehler (Error Notification)

Mitteilung von Fehlern (z.B. Zusammenbruch der Sicherungsverbindung), die von Sicherungsschicht entdeckt, aber nicht behoben werden können, an Instanz der Netzschicht

### (6) Flußkontrolle (Flow Control)

Instanzen der Netzschicht kontrollieren die Rate, mit der sie Pakete empfangen.

## Dienste der Sicherungsschicht (2)

### (7) Auswahl der Dienstgüte-Parameter (Quality of Service)

Sicherungsschicht wählt für die Dauer einer Sicherungsverbindung die Dienstgüte in Abhängigkeit der Dienstgüte auf Netzschicht) aus

#### Dienstgüte-Parameter beinhalten:

- mittlere Zeit zwischen erkannten aber nicht behebbaren Fehlern
- Restfehlerrate durch duplizierte oder verlorengegangene Pakete
- Verfügbarkeit des Dienstes, abhängig von Zuverlässigkeit der Knoten
- Übertragungsverzögerung
- Durchsatz pro Zeiteinheit von korrekt übertragenen Paketen

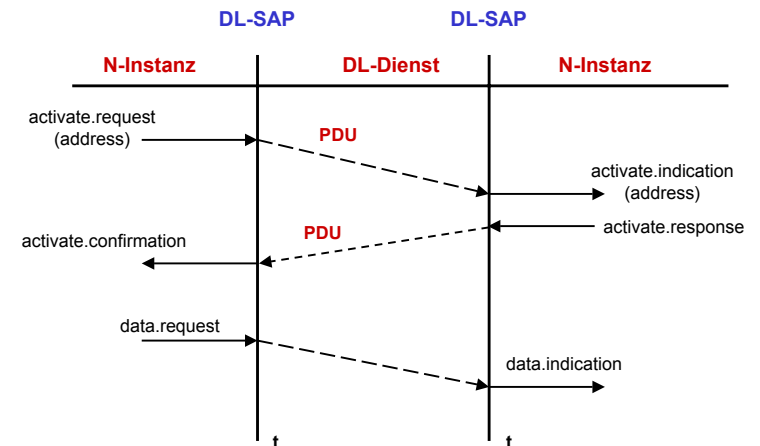


## Dienste der Sicherungsschicht

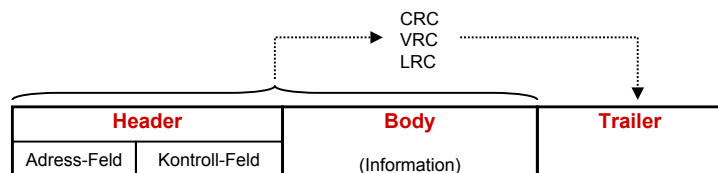
Dienst	req.	ind.	resp.	conf.	Parameter
<b>Verbind.aufbau:</b>					
connect	x	x	x	x	Adressen, Dienstgüteparameter
activate	x			x	Adressen
acquire	x			x	welche Übertragungsstrecken
negotiate	x	x	x	x	Dienstgüteparameter
<b>Transferphase:</b>					
data	x	x			Information, Adressen
expedited_data	x	x	x	x	Information, Adressen
flow control	x	x			on, off
reset	x	x	x	x	Adressen der Verbindung
notify	x	x		x	Adressen
abort	x	x			Adressen der Verbindung
<b>Verbind.abbau:</b>					
disconnect	x	x		x	Adressen
deactivate	x			x	Adressen

Primitive (x = vorhanden)

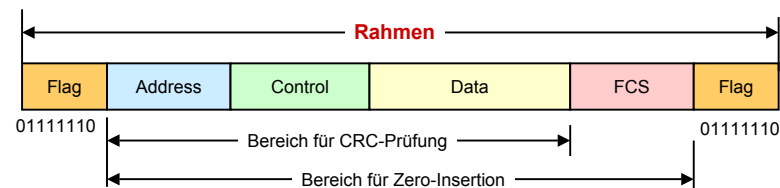
## Benutzung des Activate-Dienstes



## Allgemeines Format einer DL-PDU



### Bitorientiertes Rahmenformat



## Fehlertypen und Fehlerursachen

### Verfälschung von Bits bei der Übertragung – Bitfehler

**Beispiel:** - Null-Bit werde durch 0 Volt repräsentiert; Eins-Bit durch 5 Volt  
 - Entscheidungsschwelle sei 2,5 Volt  
 - Übertragungsstrecke ist nicht optimal: Rauschen, Signaldämpfung

**Ergebnis:** Empfänger empfängt Signalwert von 3 Volt, obwohl ursprünglich 0 Volt gesendet wurde. ⇒ Bitfehler.

### Fehlerursachen

- Rauschen
- Schaltvorgängen in benachbarten Stromkabel
- Verlust der Bit-Synchronisation

### Verfälschung von Dateneinheiten – Rahmenfehler

#### Fehlerarten

- Verlust/Duplizierung einer Dateneinheit
- Abweichung der Empfangsreihenfolge von Dateneinheiten

#### Fehlerursachen

- Bitfehler im Rahmen
- verfrühte Datenwiederholung

## Fehlertypen

### Einzelbitfehler

z.B. Rauschspitzen, die die Detektionsschwelle bei digitaler Signalerfassung überschreiten

### Bündelfehler

Länger anhaltende Störung durch Überspannung, Starkstromschaltprozesse, etc.

### Synchronisationsfehler

- Alle Bits bzw. Zeichen werden falsch erkannt
- Auswirkung einer Störung ist abhängig von der Übertragungsgeschwindigkeit

### Rechenbeispiel

#### Eine Störung von 20 ms führt

- bei Telex (50 bit/s, Signaldauer: 20 ms) zu einem Fehler von 1 Bit  $\Rightarrow$  Einzelbitfehler
- bei ISDN (64 kbit/s, Signaldauer: 15,625  $\mu$ s) zu einem Fehler von 1280 Bit  $\Rightarrow$  Bündelfehler
- bei Breitband-ISDN (mit 155 Mbit/s, Signaldauer: 6,45 ns) zu einem Fehler von ca. 3,1 Mbit

## Fehlerwirkungen

- Die Fehlerwirkungen gestörter Bits können sehr unterschiedlich sein.
- Dies ist abhängig davon welche Bits einer Dateneinheit betroffen sind.

### (Nutz-)Datenfehler

Bits innerhalb der Nutzdaten werden gestört.

### Protokollfehler

Störungen können Protokollkontrolldaten, Steuerzeichen, Adressen oder sonstige protokollrelevante Daten verfälschen oder vernichten.

Für einen zuverlässigen Kommunikationsdienst sind Mechanismen zur Fehlererkennung und -behebung erforderlich.

Diese Mechanismen tragen erheblich zur Komplexität von Protokollen bei, die einen zuverlässigen Dienst anbieten, falls sie diesen nicht bereits von unterliegenden Schichten bereitgestellt bekommen.

Kommunikationsprotokolle, die von der Annahme fehlerfreier Übertragung ausgehen, sind sehr einfach.

## Fehlererkennung und -behebung

### Bezüglich Bitfehlern

- Fehlererkennende Codes
- Fehlerkorrigierende Codes

### Bezüglich kompletter Dateneinheiten

- Sequenznummern
- Zeitüberwachung
- Quittungen
- Sendewiederholungen

### Redundanz

- Fehlererkennung von Datenfehlern beim Empfänger durch Hinzufügung von Redundanz beim Sender.

## Paritätsbits

Zu einer vorbestimmter Einheit wird jeweils ein redundantes Bit hinzugefügt

- **Gerade Parität** : es wird auf **gerade** Anzahl von 1-Bit ergänzt
- **Ungerade Parität**: es wird auf **ungerade** Anzahl von 1-Bit ergänzt

### Folgende Varianten werden unterschieden

#### Vertikale Parität

- An jedes einzelne Zeichen (bestehend aus n Bits) wird ein Paritätsbit angefügt (d.h. ein Paritätsbit pro Reihe)
- Erkennung von Bitfehlern ungerader Anzahl (1-Bitfehler, 3-Bitfehler etc.)

#### Längsparität

- An eine Folge von Zeichen wird ein dediziertes Prüfzeichen angefügt. Dieses enthält jeweils ein Paritätsbit pro Spalte (d.h. pro n-tem Bit aller Zeichen)

#### Matrixparität

- Vertikale Parität und Längsparität werden kombiniert. Jeweils 1 Paritätsbit pro Spalte und pro Reihe eines aus mehreren Zeichen bestehenden Blocks
- Auch als Block Check Character (BCC) bezeichnet.
- Falls Störungen weniger als 8 Bit betreffen (bei einer Zeichenlänge von 8 Bit), wird der Fehler gefunden.

# Cyclic Redundancy Check (CRC)

Anstelle eines einzigen Paritätsbits wird hier eine Sicherungssequenz an die zu übertragende Dateneinheit angefügt.

- Wird auch als FCS (Frame Check Sequence) bezeichnet
- Basiert auf Division in Modulo-2-Binärarithmetik; wobei keine Überträge stattfindet

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
<hr/>			
01010001	11111110	01010110	11111010

Bitstrings als Repräsentation von Polynomen, z.B. Dateneinheit 10011010 entspricht Polynom  $M(x) = x^7 + x^4 + x^3 + x$

- Dateneinheit wird als unstrukturierte Bitfolge aufgefasst, d.h. auch die Anzahl der zu prüfenden Bit ist beliebig (oberhalb einer Mindestlänge).
- Es werden auch keine ganzzahligen Vielfache von 8 Bit gefordert.

# Vorgehensweise zur Berechnung des CRC

- 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  $M(x)$
- Dateneinheit muss länger sein als Generatorpolynom
- Prüfsumme entspricht Rest R der Division  $(x^r M(x)) / G(x)$ 
  - r: Grad des Generatorpolynoms
- $x^r M(x)$  fügt r Nullstellen an das Ende der Dateneinheit

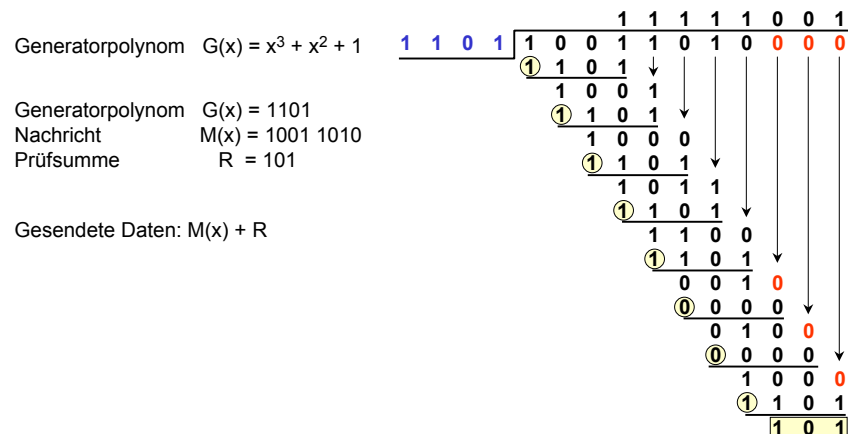
## Prüfsumme wird an die zu sendenden Daten angehängt

- Entspricht der Addition des Restes:  $x^r M(x) + R$

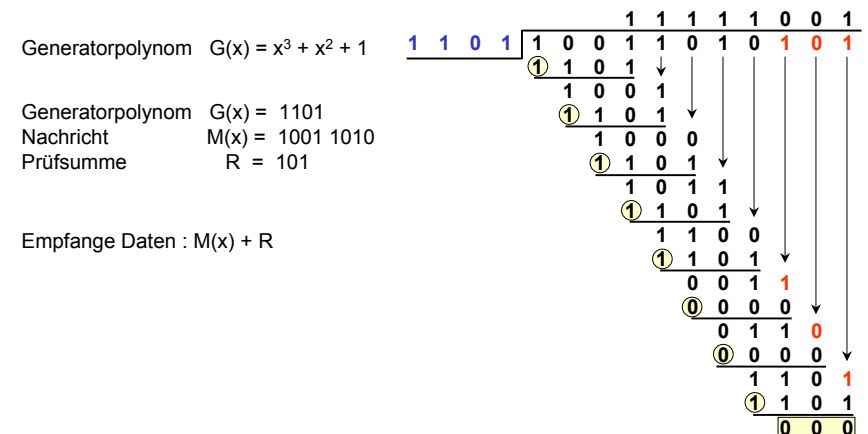
## Empfänger überprüft Dateneinheit

- Division der empfangenen Dateneinheit durch  $G(x)$ 
  - Ist der Rest der Division Null, dann wurde kein Fehler erkannt
  - Ist der Rest der Division ungleich Null, dann ist die empfangene Dateneinheit fehlerhaft

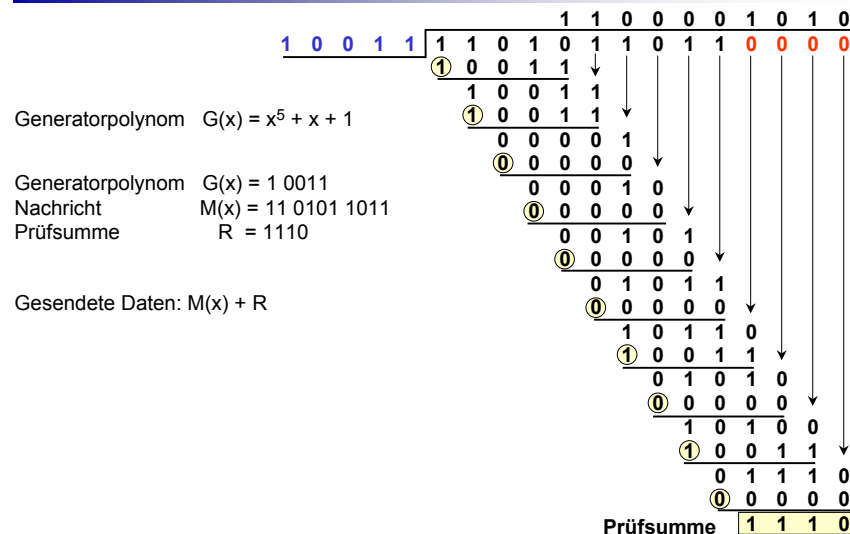
## CRC Beispiel 1: Berechnung der Prüfsumme



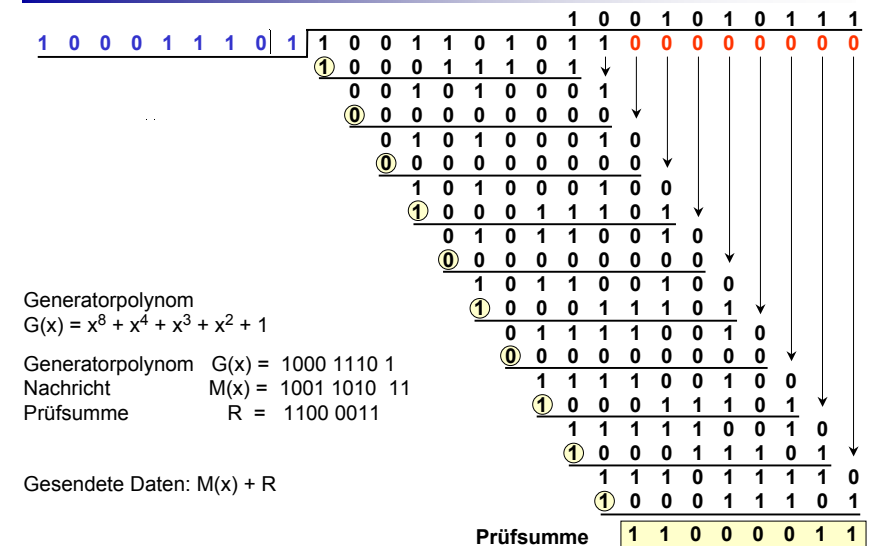
## CRC Beispiel 1: Überprüfung der Prüfsumme



## CRC Beispiel 2: Berechnung der Prüfsumme



## CRC Beispiel 3: Berechnung der Prüfsumme



## Leistungsfähigkeit von CRC

### Folgende Fehler werden durch CRC erkannt

- sämtliche Einzelbitfehler
- sämtliche Doppelfehler, wenn  $(x^k + 1)$  nicht durch das Prüfpoly nom teilbar ist, für  $k \leq$  Rahmenlänge
- sämtliche Fehler ungerader Anzahl, wenn  $(x+1)$  Faktor des Prüfpoly noms ist
- sämtliche Fehlerbursts der Länge  $\leq$  Grad des Prüfpoly noms

### International genormt sind u.a. folgende Prüfpoly nome

CRC-12 =  $x^{12} + x^{11} + x^3 + x^2 + x + 1$

CRC-16 =  $x^{16} + x^{15} + x^2 + 1$

CRC-ITU =  $x^{16} + x^{12} + x^5 + 1$

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-ITU entdecken

- alle Einzel- und Doppelfehler
- alle Fehler ungerader Anzahl
- alle Fehlerbursts mit der Länge  $\leq 16$
- 99,997 % aller Fehlerbursts mit der Länge 17
- 99,998 % aller Fehlerbursts mit der Länge 18 und mehr

## Erkennen von Bitfehlern mit CRC

### Erkennen aller Einzelbitfehler

$x^k$  und  $x^0$  dürfen nicht gleich Null sein

### Nahezu alle Doppelbitfehler

$G(x)$  muss mindestens drei Terme besitzen

### 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

## CRC-Implementierung in Hardware

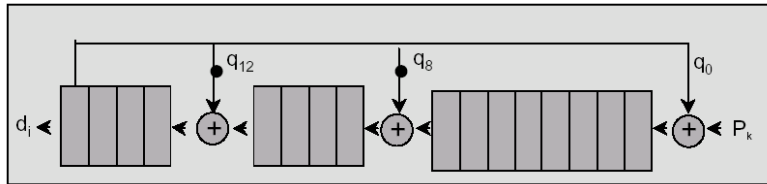
### Realisierung in Hardware

Benutzung von rückgekoppelten Schieberegistern.  
CRC kann während des Durchschiebens durch das Schieberegister berechnet werden.

### Prinzip

Dateneinheit durchläuft bitweise das Schieberegister.  
Rückkopplung erfolgt an den Stellen, an denen das Generatorpolynom auf 1 gesetzt ist.

### Beispiel



## Vorwärtsfehlerkorrektur

Bislang war die Redundanz nur zur Überprüfung der Daten mitgeliefert worden, jetzt soll sie 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 jetzt an den Empfänger gesendet.

## Vorwärtsfehlerkorrektur — Ablauf

Der Empfänger muss nur drei der vier Dateneinheiten korrekt empfangen, um die fehlende Dateneinheit rekonstruieren zu können.

Er verknüpft einfach die korrekt empfangenen Dateneinheiten mit XOR und erhält 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

Empfänger muss aber wissen, welche Dateneinheit verloren gegangen ist .

## Fehlerkontrolle bei Rahmenfehlern

Mit Paritätsbits und Prüfsummen können Bitfehler erkannt werden.

### Erkennung

Zur **Erkennung** von Fehlern bezüglich kompletter Dateneinheiten (Rahmenfehler) sind zusätzliche Mechanismen erforderlich.

- Sequenznummern (Sequence number)
- Zeitgeber (Timer)

Auch wenn alle Dateneinheiten empfangen wurden, können die genannten Mechanismen erforderlich sein.

### Behebung

Zur **Behebung** der Fehler werden die folgenden Mechanismen verwendet

- Quittungen (Acknowledgements)
- Sendewiederholungen (Retransmissions)

# Sequenznummern

## Problemstellung

- Woher weiß der Empfänger, ob die Dateneinheiten in der richtigen Reihenfolge ankommen?
- keine Duplikate enthalten sind?
- keine Dateneinheiten fehlen?

## Mechanismus

- Die Dateneinheiten (oder die Bytes) werden durchnummeriert.
- Eine entsprechende Kennung wird mit jeder Dateneinheit übertragen.
- Diese Kennung wird als *Sequenznummer* bezeichnet.

## Fehlerszenarien

Sequenznummern helfen, wenn Dateneinheiten nicht ausgeliefert werden.

# Zeitgeber

## Problemstellung

Wann entscheidet ein Empfänger, dass eine Dateneinheit nicht angekommen ist?

## Mechanismus

In Abhängigkeit einer zeitlichen Obergrenze wird *vermutet*, dass eine Dateneinheit nicht mehr beim Empfänger eintrifft.

Der Empfänger startet einen Zeitgeber jeweils dann, wenn er eine korrekte Dateneinheit empfangen hat. Wird die nächste nicht innerhalb dieses so vorgegebenen Zeitintervalls empfangen, so wird vermutet, dass sie im Netz verloren ging.

# Quittungen (1)

## Problemstellung

Wie erfährt der Sender, dass eine Dateneinheit überhaupt nicht bzw. nicht korrekt beim Empfänger angekommen ist?

## Mechanismus

- Der Empfänger teilt dem Sender mit, ob er eine Dateneinheit empfangen hat oder nicht.
- Hierzu werden spezielle Dateneinheiten, sogenannte Quittungen, versendet (ACK: Acknowledgement).

## Varianten

### Positive Quittung

Empfänger teilt dem Sender mit, dass er die entsprechenden Daten erhalten hat.

### Negative Quittung

- Empfänger meldet dem Sender, dass er die entsprechenden Daten nicht erhalten hat (z.B. bei falscher Reihenfolge).

NACK: Negative Acknowledgement

Quittungen werden oftmals in Kombination mit Zeitgebern verwendet.

# Quittungen (2)

Weitere Varianten

## Selektive Quittungen

Die Quittung bezieht sich auf eine einzelne Dateneinheit.

## Beispiel

- Negative selektive Quittung, falls der Verlust einer Dateneinheit vom Empfänger vermutet wird (NACK).
- Lässt sich mit NACKs ein zuverlässiger Dienst bereitstellen?  
SACK: Selective Acknowledgement

## Kumulative Quittungen

Die 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.



# Automatic Repeat Request: ARQ

## Grundlegende Variante zur Sendewiederholung

- Sender erhält positive Quittungen über den Erhalt einer Dateneinheit.
- Sender kann Sendewiederholungen ausführen.

## Varianten

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

## Situation

Eine Reihe verschiedener ARQ-Mechanismen ist verfügbar, wobei zwischen grundlegenden Unterschieden und Implementierungseinheiten differenziert werden muss. Im folgenden werden grundlegende Varianten diskutiert.

# Send-and-Wait

## Grundlegendes einfaches ARQ-Verfahren

Sender wartet auf die Quittung zu einer gesendeten Dateneinheit, bevor er eine neue Dateneinheit senden darf.

- Falls keine Quittung empfangen wird, erfolgt die Wiederholung der Dateneinheit nach dem Ablauf des Zeitgebers.
- Es können Duplikate von Dateneinheiten entstehen.

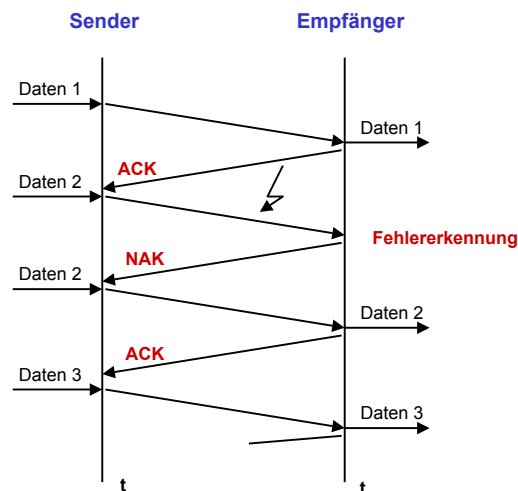
## Sequenznummern

- Wie viele Bits werden bei Stop-and-Wait für die Sequenznummer benötigt?
- Annahme: Sequenznummern identifizieren Dateneinheiten.

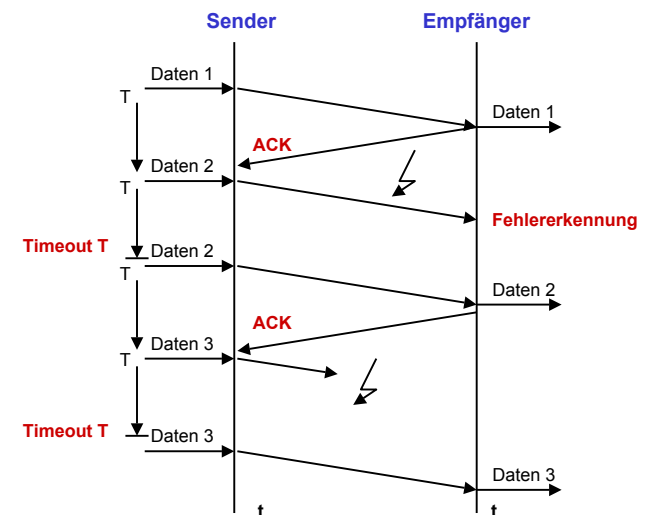
## Nachteil

Stets nur eine nicht quitierte Dateneinheit beim Sender

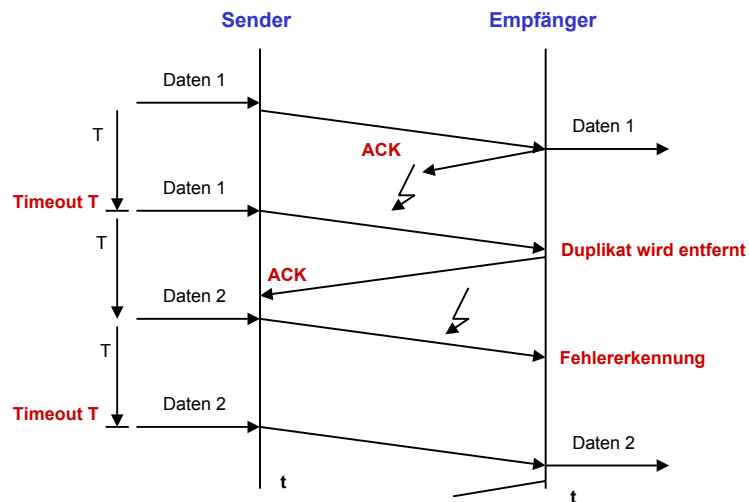
# Send-and-Wait



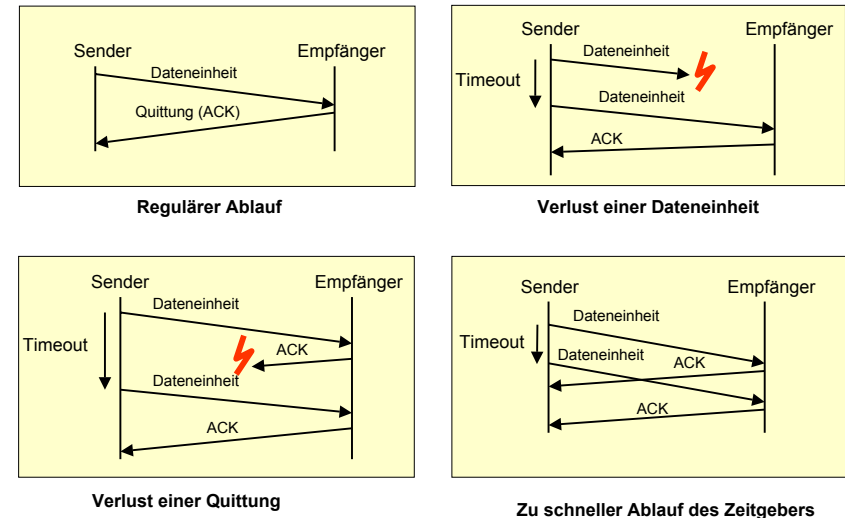
# Send-and-Wait



## Send-and-Wait



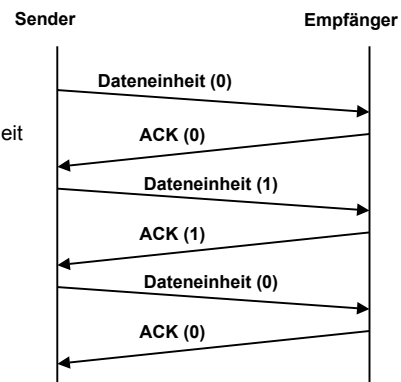
## Send-and-Wait : Szenarien



## Send-and-Wait : Sequenznummern

### Problem

In den vorangegangenen Szenarios 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).

## Go-Back-N ARQ

### Ziel

Erhöhung des Durchsatzes im Vergleich zu Stop-and-Wait. Das Warten auf eine Quittung vor dem Senden der nächsten Dateneinheit soll vermieden werden.

### Verfahren

Der Sender kann mehrere Dateneinheiten senden bis er eine Quittung erhalten muss. Die maximale Anzahl der nicht quittierten Dateneinheiten ist begrenzt (typischerweise durch ein sogenanntes Sliding Window).

### Variante 1

- Empfänger quittiert korrekt empfangene Dateneinheiten wie bei Stop-and-Wait.
- Die Quittung kann allerdings kumulativ erfolgen, d.h. für mehrere Dateneinheiten auf einmal kumulative Sequenznummer gibt an, bis wohin die Daten korrekt empfangen wurden, d.h. es handelt sich um positive Quittungen.
- Alle noch nicht quittierten aber gesendeten Daten werden vom Sender wiederholt (Go-Back-N, wobei N die kumulative Sequenznummer ist)

### Variante 2

- Nicht korrekt empfangene Dateneinheiten werden mit einer negativen Quittung (NACK) bestätigt
- Sender wiederholt daraufhin ab dieser Sequenznummer alle gesendeten Dateneinheiten (Go-Back N, wobei N die Sequenznummer in der negativen Quittung ist).

## Selective Reject ARQ

### Ziel

Erhöhung der Datenrate im Vergleich zu Stop-and-Wait. Reduzierung des Datenaufkommens im Vergleich zu Go-Back-N.

### Verfahren

- Der Sender kann mehrere Dateneinheiten senden, bis er eine Quittung erhalten muss.
- Die maximale Anzahl der nicht quitierten Dateneinheiten ist begrenzt.  
(wie bei Go-Back-N)
- Der Empfänger sendet eine negative Quittung, wenn er einen Fehler erkennt.
- Diese Quittung bezieht sich auf eine einzelne Dateneinheit.
- Empfänger wiederholt genau die Dateneinheit mit der bei der negativen Quittung angegebenen Sequenznummer.
- Nur die nicht korrekt empfangenen Dateneinheiten werden vom Senderwiederholt.

## Flusskontrolle mit Halt-/Weiter-Meldungen

### Einfachste Methode

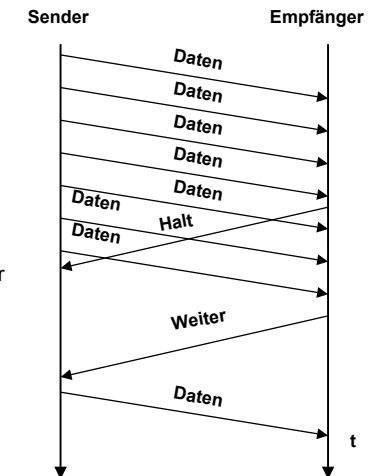
Sender-Empfänger-Flusskontrolle

### Meldungen

- Halt
- Weiter

Kann der Empfänger nicht mehr Schritt halten, schickt er dem Sender eine Halt-Meldung.

Ist ein Empfang wieder möglich, gibt der Empfänger die Weiter-Meldung.



## Implizite Flusskontrolle

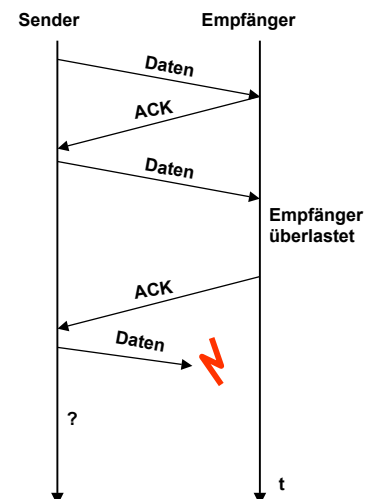
### Funktionsweise

Durch Zurückhalten der Quittung (z.B. ACK/NACK) kann der Sender gebremst werden.

Das bedeutet, dass ein Verfahren zur Fehlererkennung für die Flusskontrolle mitbenutzt wird.

### Problem

Der Sender kann nicht mehr unterscheiden, ob seine Dateneinheit völlig verloren ging ob der Empfänger die Quittung wegen Überlast zurückgehalten hat.

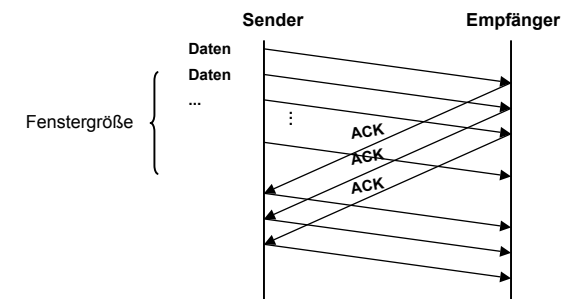


## Kreditbasierte Flusskontrolle

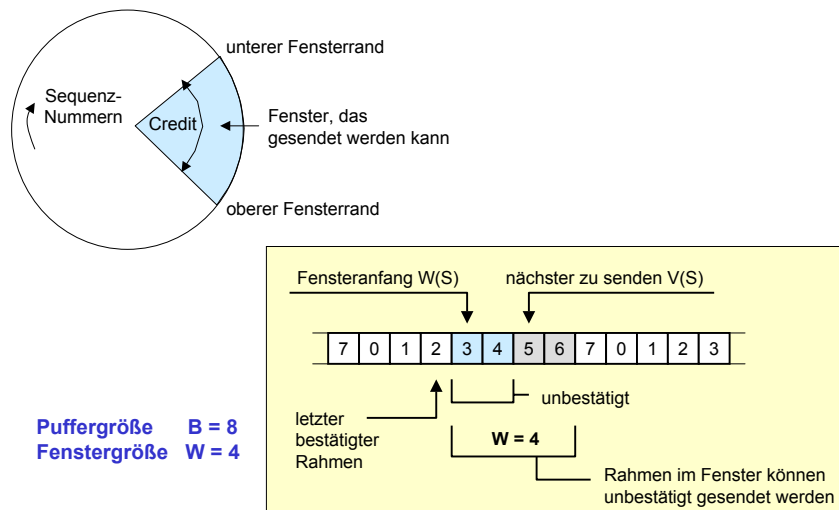
### Prinzip

Sender kann bis zu einer maximalen Anzahl Dateneinheiten senden, ohne eine Quittung zu empfangen.

- Maximale Anzahl der Dateneinheiten repräsentiert die Pufferkapazität des Empfängers und wird als Sendekredit bezeichnet.
- Oftmals als fortlaufendes Fenster bezeichnet (Sliding Window)
- Fenster wird dann mit jeder empfangenen positiven Quittung weitergeschaltet.
- Empfänger kann meist zusätzlich den Kredit explizit bestimmen.

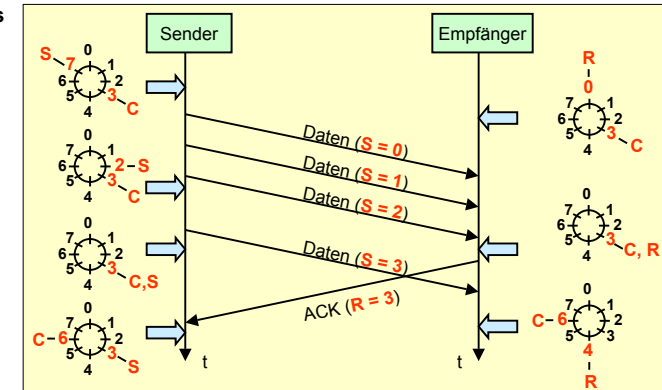


## Fenstermechanismus



## Kreditbasierte Flusskontrolle

### Fenstermechanismus (Kredit 4)



**Nachteil:** Kopplung von Fluss- und Fehlerkontrolle.

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

## Flusskontrolle mit Sliding Window

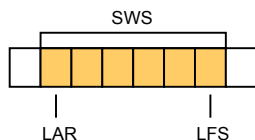
### Sender

**SWS: Send Window Size**  
(max. Anzahl ausstehender Dateneinheiten)

**LAR: Last ACK Received**  
(Sequenznummer der letzten quitierten Dateneinheit)

**LFS: Last Frame Sent**  
(Sequenznummer der letzten gesendeten Dateneinheit)

$$SWS = LFS - LAR + 1$$



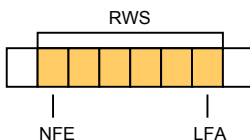
### Empfänger

**RWS: Receiver Window Size**  
(max. Anzahl nicht in Reihenfolge empfangener Dateneinheiten)

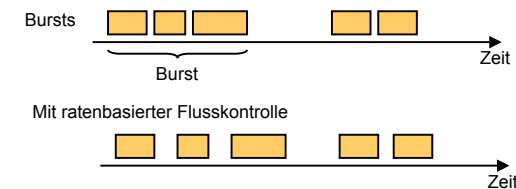
**LFA: Last Frame Accepted**  
(Sequenznummer der letzten empfangbaren Dateneinheit)

**NFE: Next Frame Expected**  
(Sequenznummer der nächste in Reihenfolge erwartete Dateneinheit)

$$RWS = LFA - NFE + 1$$



## Ratenbasierte Flusskontrolle



### Prinzip

Sender kann Daten mit einer nach oben begrenzten Rate senden, d.h. es existiert ein minimales Zeitintervall zwischen aufeinanderfolgenden Dateneinheiten, das nicht unterschritten werden darf.

Folgende Parameter sind relevant

- Zeitintervall zwischen zwei aufeinanderfolgenden Dateneinheiten
- Maximale Größe der Dateneinheiten

### Vorteil

Vermeidet Bursts (Deutsch: Büschel)

# Automaten der Sicherungsinstanz

