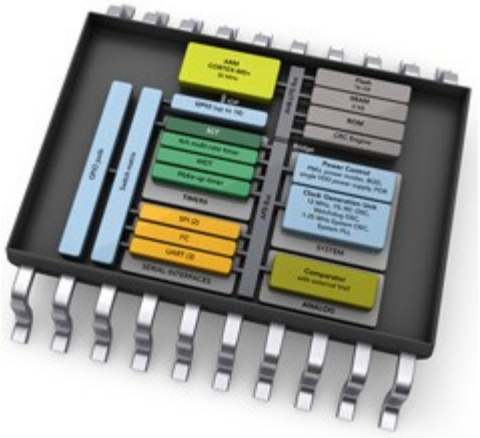
FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Kommunikationssysteme

(Modulcode 941306)

Prof. Dr. Andreas Terstegge



Sliding Window Verfahren - Schiebefensterverfahren

Alle auf einmal

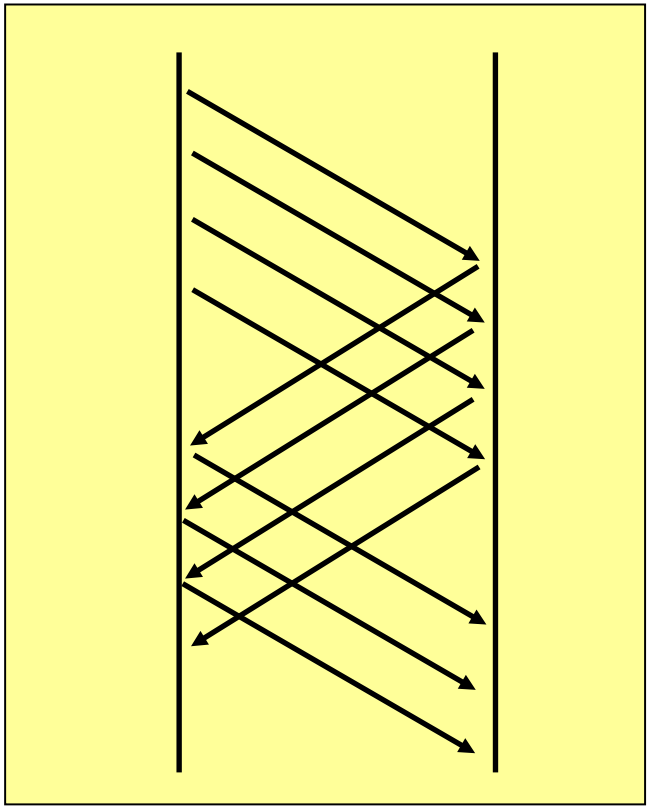
- Segmente müssen auf Sender- und Empfängerseite gepuffert werden
  - Sender muss ein Segment speichern, bis es bestätigt wurde
  - Empfänger muss ein Segment speichern, bis die Anwendung es ausliest und verarbeitet
  - Senden aller Segmente auf einmal:
    > Große Puffer im Betriebssystem nötig
    > Netz wird spontan sehr stark belastet

- **Prinzip der Flusskontrolle**
  - Erlaube das Senden mehrerer (n) Segmente ohne Quittierung
  - Der Empfänger kann den Wert von n an seinen Puffer anpassen
  - Die aktuelle Netzauslastung wird auch mit berücksichtigt

Die Idee des Sliding-Window-Protokolls

Pipelining:

- Senden ohne vorheriges ACK
- Fensterbreite beim Sender: Anzahl der Segmente, die ohne Bestätigung gesendet werden
- Fensterbreite beim Empfänger: Anzahl der Segmente, die auch bei Lücken oder Fehlern zwischengespeichert werden



Fensterbreite und Segment- bzw. Sequenznummern

- Verwendung eines „Fensters“ der Größe W (in Segmenten oder in Bytes)
  - Sender und Empfänger vereinbaren ein **Übertragungsfenster** (= Größe des freien Pufferspeichers)
  - Sender nummeriert die zu versendenden Bytes/Segmente des Datenstroms z.B. mit 0, 1,..., MODULUS-1, 0, … durch, wobei *W* <= MODULUS gelten muss

    > Fenstergröße *W* bedeutet: der Sender darf maximal *W* fortlaufend nummerierte Bytes/Segmente verschicken, ohne eine Bestätigung zu bekommen

    > Empfänger bestätigt empfangene Bytes durch Quittungen (ACK)

    > Sender „rückt“ Fenster um die Anzahl der Segmente/Bytes vor, sobald ein ACK eintrifft, und darf neue Segmente verschicken

Das Prinzip der Sliding-Window-Protokolle

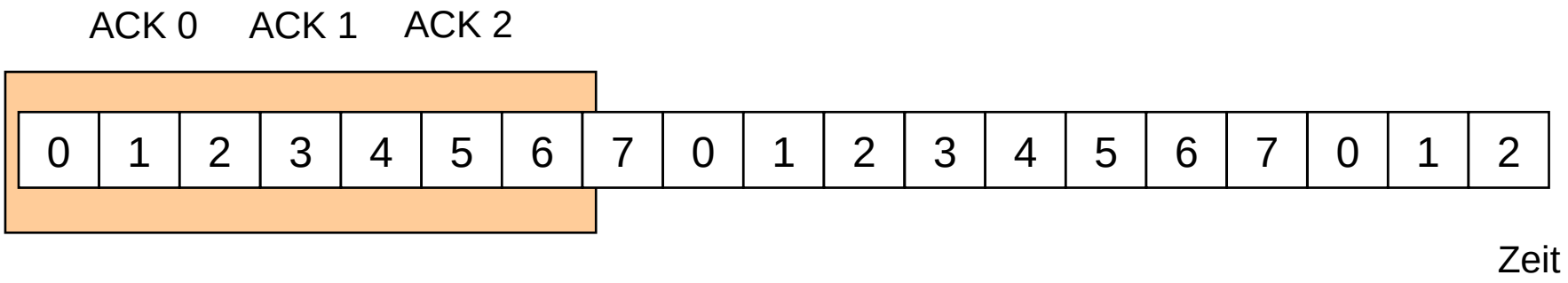
- Jedes Segment wird durch eine im Transport-Header gespeicherte Sequenznummer durchlaufend gekennzeichnet
  - Zyklische Nummerierung der Segmente

    (z.B. 0, 1, 2, ..., *MODULUS*-1, 0, ...; wobei $W < MODULUS$)
  - Wir können alternativ auch die Position des ersten Bytes des Segments im Stream nehmen, besser bei variabel langen Segmenten
  - Damit gibt es die Fenstergrenzen MIN, MAX mit
    - W = MAX – MIN + 1 für MIN < MAX und
    - W = MODULUS – MIN + MAX + 1 für MIN > MAX
- Empfänger bestätigt den korrekten Empfang durch Quittungen (*ACK*)
- ACKs beziehen sich auf eine Sequenznummer
  - Nummer des empfangenen Segments
  - Nummer des als nächstes erwartetes Segment
  - Offset des als nächstes erwartetes Segment im Stream
- Übertragungsfenster des Senders wird bei Empfang eines ACK angepasst (MIN und MAX werden angepasst), was faktisch ein Verschieben des Fensters bedeutet

Sliding-Window-Protokoll

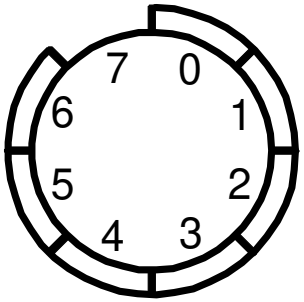
Beispiel (3-Bit Sende-/Empfangsnummer – MODULUS = W+1)

- Der Einfachheit halber nummerieren wir Segmente durch
- Sequenznummern mit 3 Bits: *m* = 8 mögliche Kombinationen
- Übertragungsfenster muss **kleiner** sein, z.B. können die Stationen eine Fenstergröße *W=7* mit 1 $W < m$ vereinbaren
- Das Fenster beschränkt die Anzahl der unbestätigten Byte, die zu einem Zeitpunkt unterwegs sein dürfen (hier max. 7 wegen *W* = 7)
- Bei Empfang einer Quittung wird das Fenster entsprechend verschoben
- Bytes werden fortlaufend modulo *m* nummeriert (für *m* = 8 also Nummern von 0 bis 7)

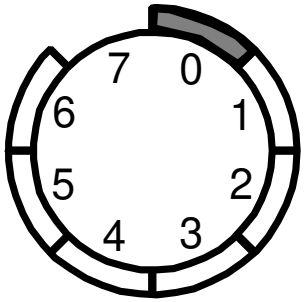


Sliding Window - andere Darstellung

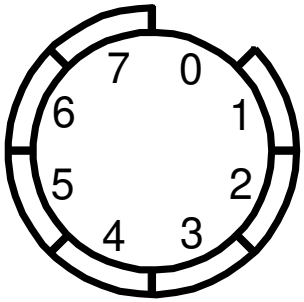
- Beispiel (für 3-Bit Sende-/Quittungsnummer)
  - 3 Bit Sende-/Quittungsnummer → $m = 8$ mögliche Kombinationen
  - Bytes werden fortlaufend modulo m nummeriert
  - Stationen vereinbaren Fenstergröße W mit $1 \leq W < m$, z.B. $W = 7$
  - W beschränkt die Anzahl der unbestätigten Bytes, die zu einem Zeitpunkt unterwegs sein dürfen (hier max. 7)
  - Bei Empfang einer Quittung wird Fenster entsprechend verschoben



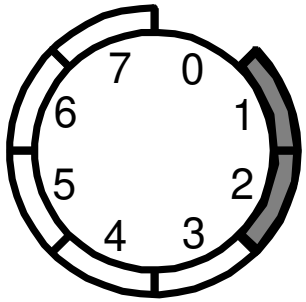
Station sendet Byte 0 - 6



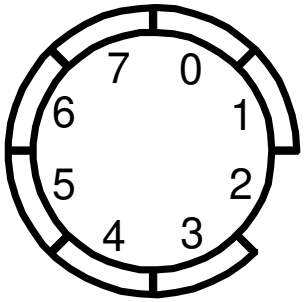
Station erhält Quittung für Byte 0 (d.h. ACK 1)



Station verschiebt Fenster um 1, sendet Byte 7



Station erhält Quittung für 1, 2



Station verschiebt Fenster um 2, sendet Byte 0, 1

Wie groß sollte das Übertragungsfenster sein?

Basis: Berechnung des Nutzungsgrades aus letzter Vorlesung

$$\rho = \#seg * (L/R) / (L/R + RTT)$$

$$R\rho = \#seg * L / (L/R + RTT)$$

$R\rho$ = erzielbare Bandbreite = **B**

#seg * L = Fensterbreite in bit = **W**

Es gilt L/R << RTT (siehe vorherige Beispiele)

B <= W / RTT bzw. **W >= B * RTT**

Wie groß sollte das Übertragungsfenster sein?

Das Bandwidth-Delay-Produkt als Grundgesetz!

- Sei w die Größe des Übertragungsfenster in **Bits**:
  - Wir übertragen maximal ein Fenster pro RoundTrip
  - Erreichbarer Durchsatz ≤ w / RTT
  - Bandwidth-Delay-Produkt: w >= Bandwidth * Delay

Beispiel 1:

- Die Anwendung möchte eine Rate von 10Mbps erreichen. Wir kennen die Round-Trip-Zeit, die bei 8ms liegt. Die Segmente seien 500 Byte groß. Wie groß muss w sein?
  - > Antwort: w >= 10 Mb/s * 0,008s = 80.000 Bit = 10.000 Byte
  - > In Segmenten ausgedrückt: 20 Segemente a 500 Byte

Wie groß sollte das Übertragungsfenster sein?

Das Bandwidth-Delay-Produkt als Grundgesetz!

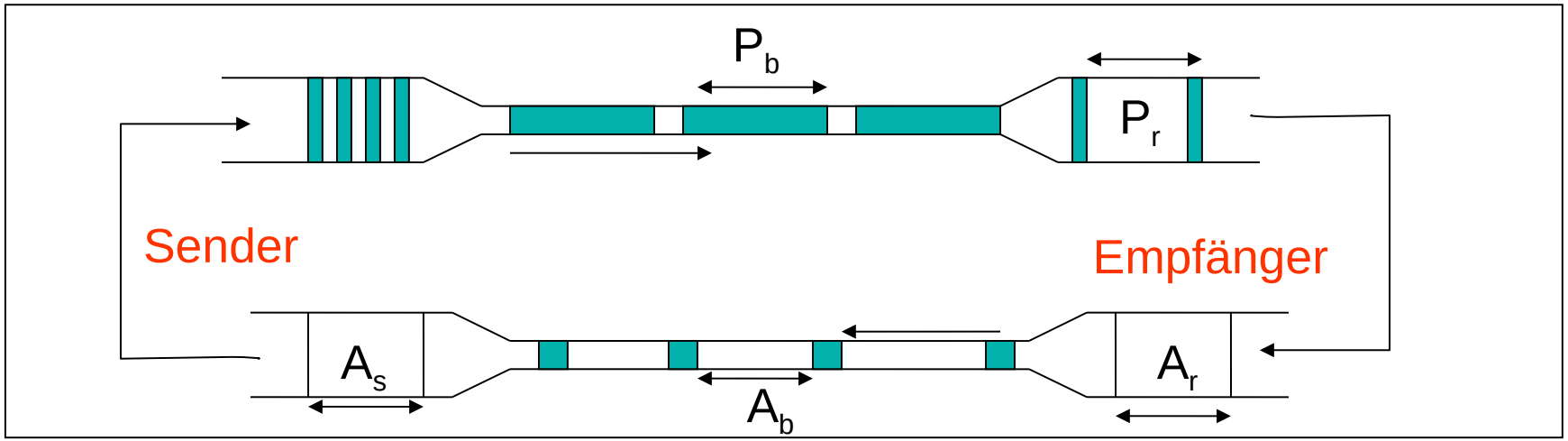
- Sei w die Größe des Übertragungsfenster in **Bits**:
  - Wir übertragen maximal ein Fenster pro RoundTrip
  - Erreichbarer Durchsatz ≤ w / RTT
  - Bandwidth-Delay-Produkt: w >= Bandwidth * Delay

Beispiel 2:

- Die Anwendung möchte eine Rate von 1Gbps erreichen. Wir kennen die Round-Trip-Zeit, die bei 240ms liegt. Die Segmente seien 1000 Byte groß. Wie groß muss w sein?
  - > Antwort: w >= 1 Gb/s * 0,24s = 240.000.000 Bit = 30.000.000 Byte
  - > In Segmenten ausgedrückt: 30.000 Segmente a 1.000 Byte

Sliding Window liefert Flusskontrolle

- Feedback-Informationen zwischen Empfänger und Sender reguliert die Senderate der Quelle
- Flusskontrolle
- Der Fortschritt des Sliding-Windows passt sich den Fähigkeiten des Netzes an
- Im „*steady state*“ wird immer dann ein Paket verschickt, wenn ein ACK empfangen wird
  - Gleichgewicht stellt sich ein



Verfahren bei Fehler/Sonderfällen

Welche Ereignisse können den geschilderten Ablauf stören?

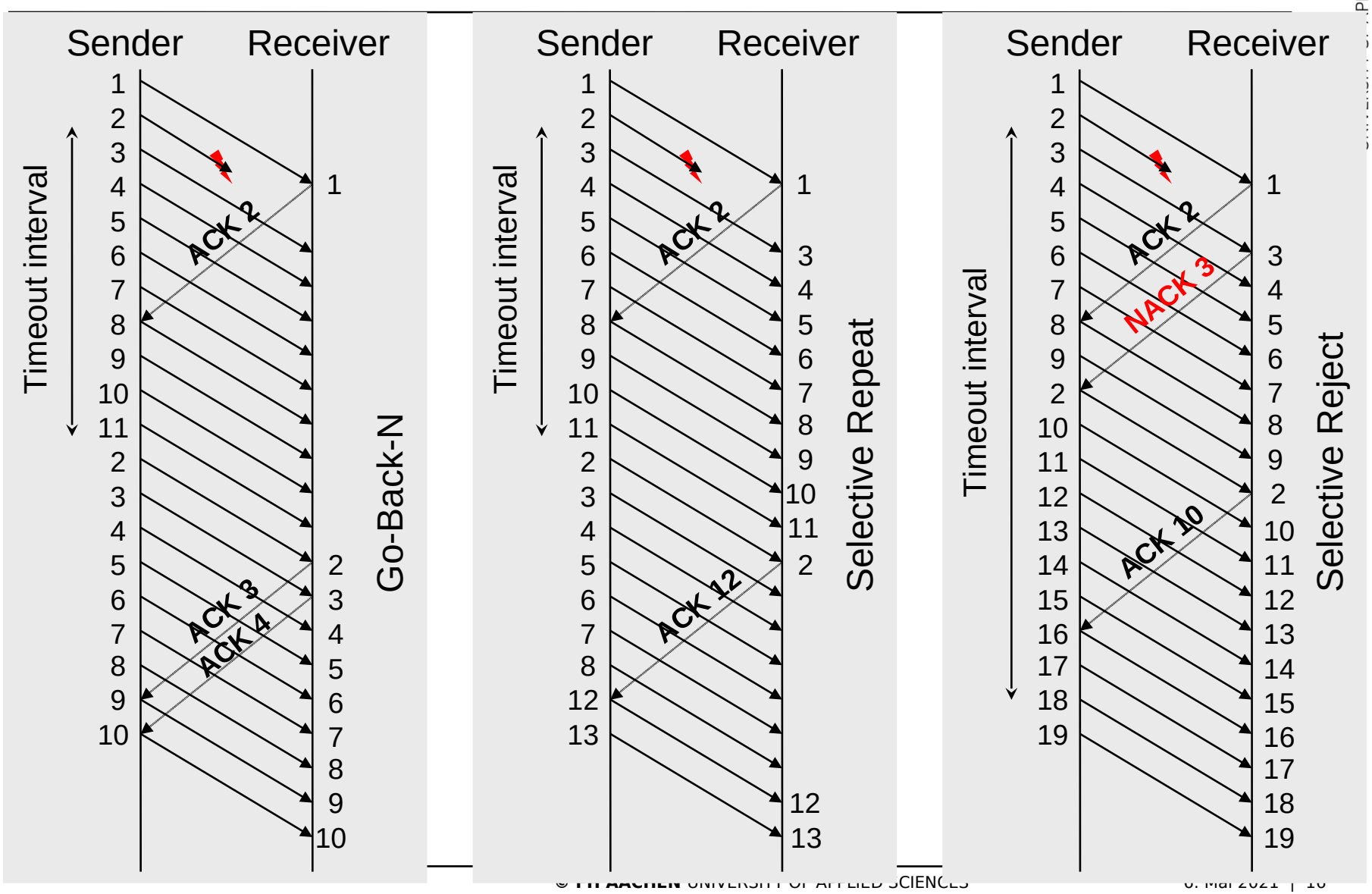
Ereignisse

- beim Empfänger:
  - Empfang eines out-of-order-Segments
    - > Eigentlich wurde ein anderes Segment erwartet
    - > Je nach Implementierung wird das nicht erwartete Segment einfach ignoriert, es wird dann verworfen
  - Empfang eines fehlerbehafteten Segments
    - > Keine Reaktion (Der Sender wird irgendwann eine erneute Übertragung vornehmen, da ja keine Bestätigung verschickt wurde)
    - > Falls unterstützt: Verschicken einer „negativen“ Bestätigung oder einer Nachricht, die dies kommuniziert (als Indiz für einen möglichen Paketverlust)

Ereignisse

- beim Sender:
  - Timeout -> Worauf bezieht sich dieser?
  - Empfang einer nicht erwarteten Bestätigung (out-of-order) ACKs
  - Empfang einer negativen Bestätigung(falls es die gibt)
  - Reaktion
    - Identifikation und erneutes Verschicken des Segments
      - **(Go-Back-N)**: Keine Kenntnis über etwaig bereits empfangene Segmente, wir übertragen ab dem Fehlerfall alles neu; Empfänger ignoriert nicht erwartete Segmente
      - **(Selective Repeat)**: Mittelbare Kenntnis über etwaig bereits empfangene Segmente, Empfänger puffert nicht erwartete Segmente, so dass sich bei Neuübertragung der „Lücke“ das Fenster ruckartig verschiebt
      - **(Selective Reject)**: Unmittelbare Kenntnis über unerwartete Ereignisse und sofortige Reaktion zur Behebung
  - Anmerkung: Alle Segmente müssen bis zum Empfang einer Bestätigung beim Sender im Betriebssystem gespeichert bleiben

Flusskontrolle: Varianten



Kumulative Bestätigungen als Kompromis

Idee der kumulativen Bestätigungen:

Eine Bestätigung zeigt an, dass alle Daten bis zu der verwendeten Nummer/Byte Position korrekt empfangen wurden

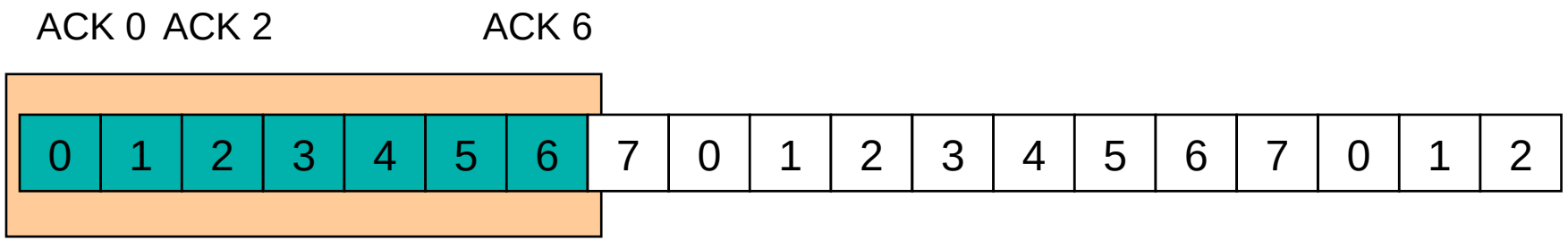
Kommt ein Out-of-Order-Segment an, so wird demnach eine Bestätigung verschickt, die schon einmal verschickt wurde (“ich warte immer noch auf x“)

Speichert der Empfänger außerhalb der Reihenfolge empfangene Segmente zwischen, dann bedeutet ein „Lückenschluss“, dass dieser einen „Sprung“ bei den Bestätigungen bedingt

Sliding-Window-Protokoll mit kumulativen ACKs

Beispiel (3-Bit Sende-/Empfangsnummer mit MODULUS = W+1)

- Bestätigungen werden kumuliert verschickt werden
- Bei Empfang einer Quittung wird das Fenster entsprechend verschoben
- Bytes werden fortlaufend modulo *m* nummeriert (für *m* = 8 also Nummern von 0 bis 7)



Kumulative Bestätigungen als Ersatz für NACKs

Wir wissen:

Kommt ein Out-of-Order-Segment an, so wird demnach eine Bestätigung verschickt, die schon einmal verschickt wurde (“ich warte immer noch auf x“)

Wenn ein Segment verloren gegangen ist, dann werden alle weiteren empfangenen Segmente die gleiche Position anzeigen

Duplicate Acknowledgments: Empfang einer Bestätigung, die eine schon versendetes Feeback erneut gibt: (“ich warte immer noch auf x“)

Segmente können sich überholen, deshalb ist das nicht sofort als NACK aufzufassen

Aber: **Triple Duplicate Acknowledgement wird als NACK aufgefasst!**

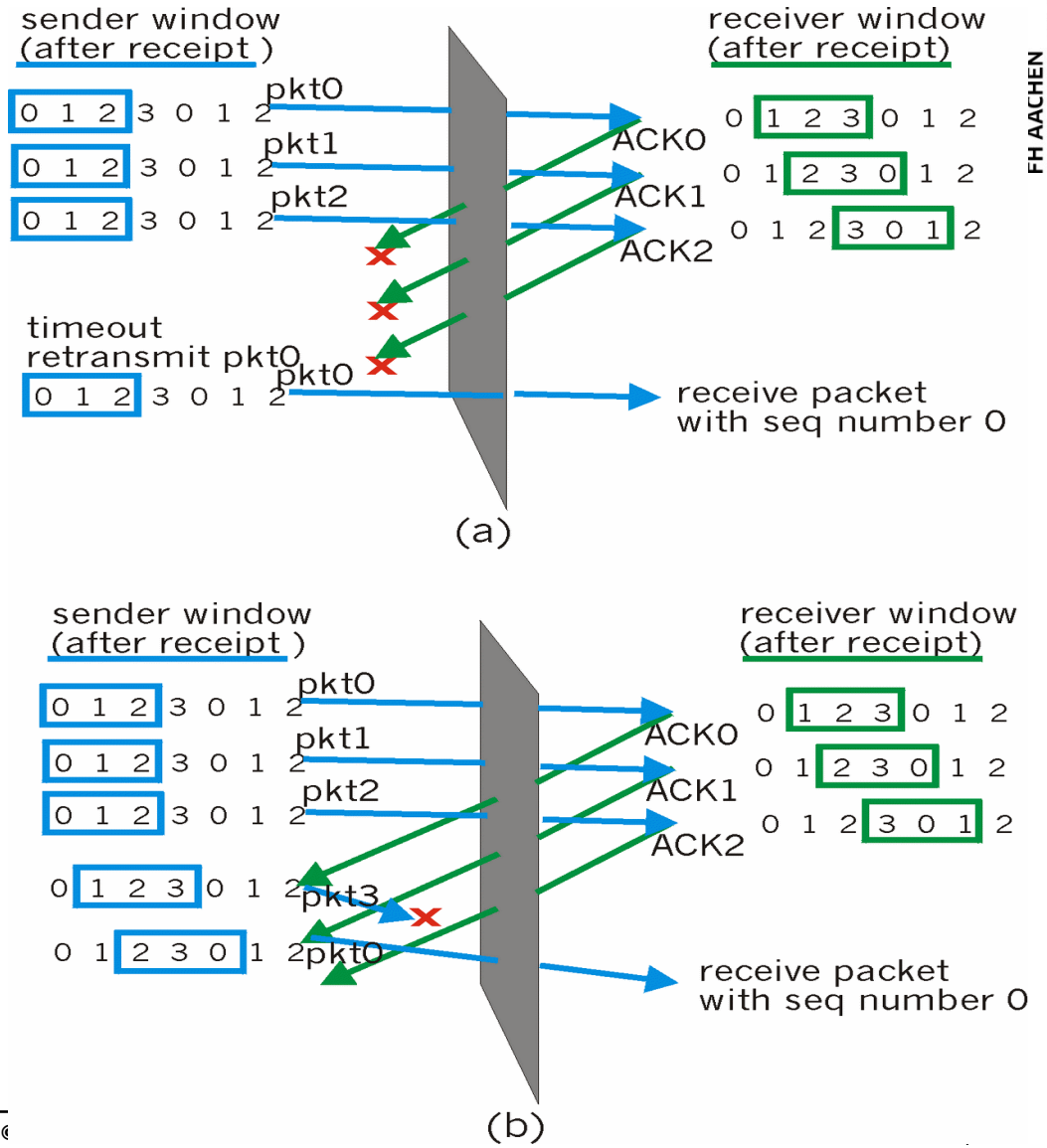
Selective Repeat und Fensterbreite

Beispiel:
Sequenznummern: 0, 1, 2, 3
Fenstergröße = 3

Empfänger sieht keinen Unterschied zwischen den beiden Szenarios!
Fehlerhaftes Versenden neuer Daten in a

→ Bei selective Repeat darf die Fensterbreite w maximal MODULUS/2 sein!

In diesem Beispiel also 2



FH Aachen
Fachbereich 9 Medizintechnik und Technomathematik
Prof. Dr.-Ing. Andreas Terstegge
Straße Nr.
PLZ Ort
T +49. 241. 6009 53813
F +49. 241. 6009 53119
Terstegge@fh-aachen.de
www.fh-aachen.de