Inhalt der Vorlesung „Rechnerkommunikation“

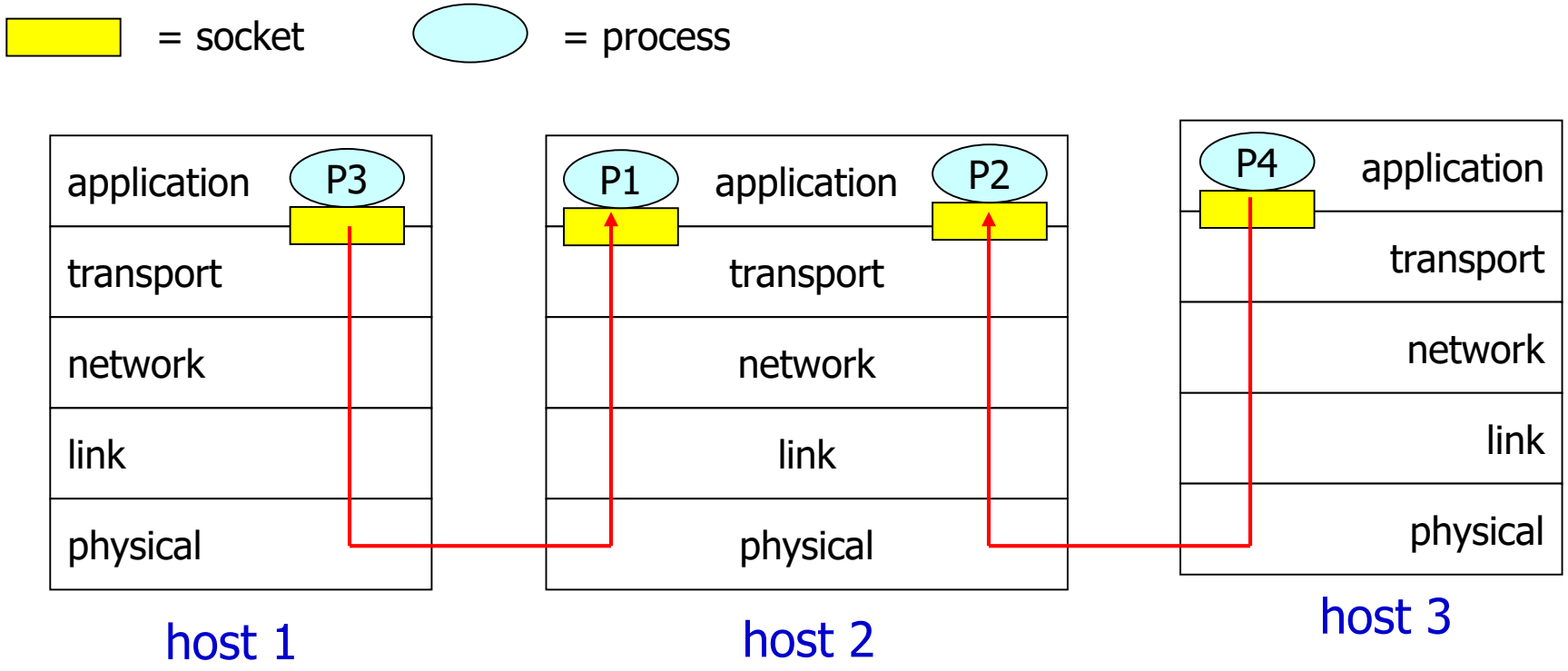
- Einführung
- Anwendungsschicht
- Transportschicht
- Netzwerkschicht
- Sicherungsschicht
- Physikalische Schicht

Transportschicht

- Einführung
- UDP
- Fehlerkontrolle
- TCP
- TLS
- QUIC

Einführung

- Aufgabe der Transportschicht: Bereitstellen eines Dienstes zur Kommunikation zwischen Anwendungsprozessen



Einführung

- mögliche Dienstmerkmale
  - Fehlerkontrolle
  - Bewahrung der Reihenfolge
  - verbindungslos/verbindungsorientiert
  - Fluss- und Überlastkontrolle
  - Garantien für Dienstgüte (z.B. Bitrate, Verzögerung, Verlust)

Fluss schützt das Zielsystem, Überlast das Netzwerk

TCP und UDP haben das nicht

- User Datagram Protocol (UDP)
  - verbindungslos, keine Kontrollmechanismen, bewahrt nicht Reihenfolge
  - Schnittstelle für einfache Paketvermittlung mittels IP, Verantwortung für Kontrollmechanismen bei Anwendung

Fehlerkontrolle existiert, wird meist aber nicht verwen

Sprich as barebones as possible

- Transmission Control Protocol (TCP)
  - verbindungsorientiert, Fehler-, Fluss-, Überlastkontrolle, keine Dienstgütegarantien
  - bietet Abstraktion eines Bytestroms

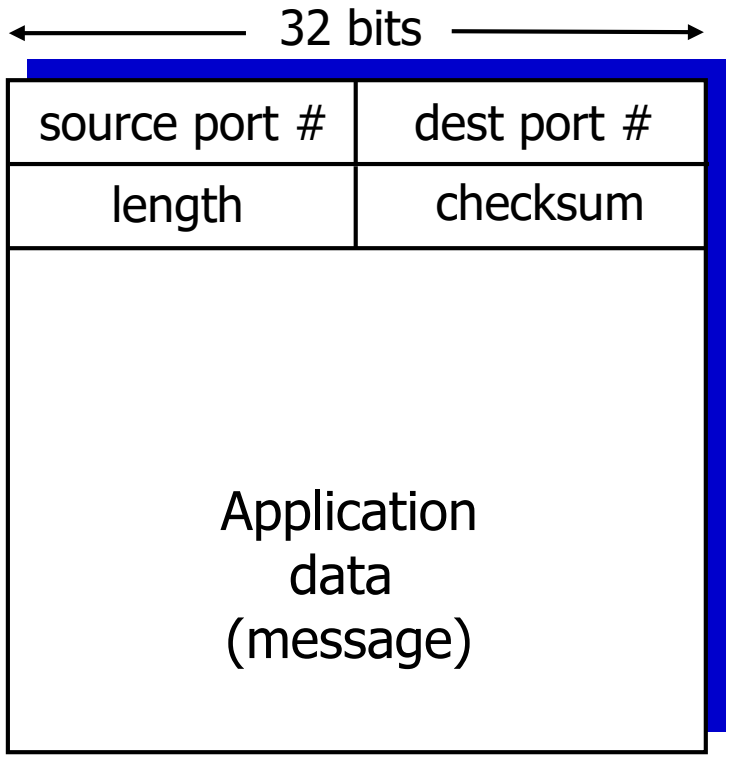
Transportschicht

- Einführung
- UDP
- Fehlerkontrolle
- TCP
- TLS
- QUIC

UDP

- Segment:
  - source port:
    Quellportnummer (16 Bit)
  - dest port:
    Zielportnummer (16 Bit)
  - length:
    Länge des gesamten Segments (16 Bit)
  - checksum:
    Prüfsumme (16 Bit) für mögliche Fehlerkontrolle,
    Benutzung ist optional,
    0000000000000000_2 bedeutet: ungenutzt
  - Frage: wo befinden sich Quell- und Ziel-IP-Adresse?

IP-Schicht



Quelle: Kurose, Ross. *Computer Networking: A Top-Down Approach*, 7th Ed., Pearson Education, 2017.

UDP

Multiplexen und Demultiplexen

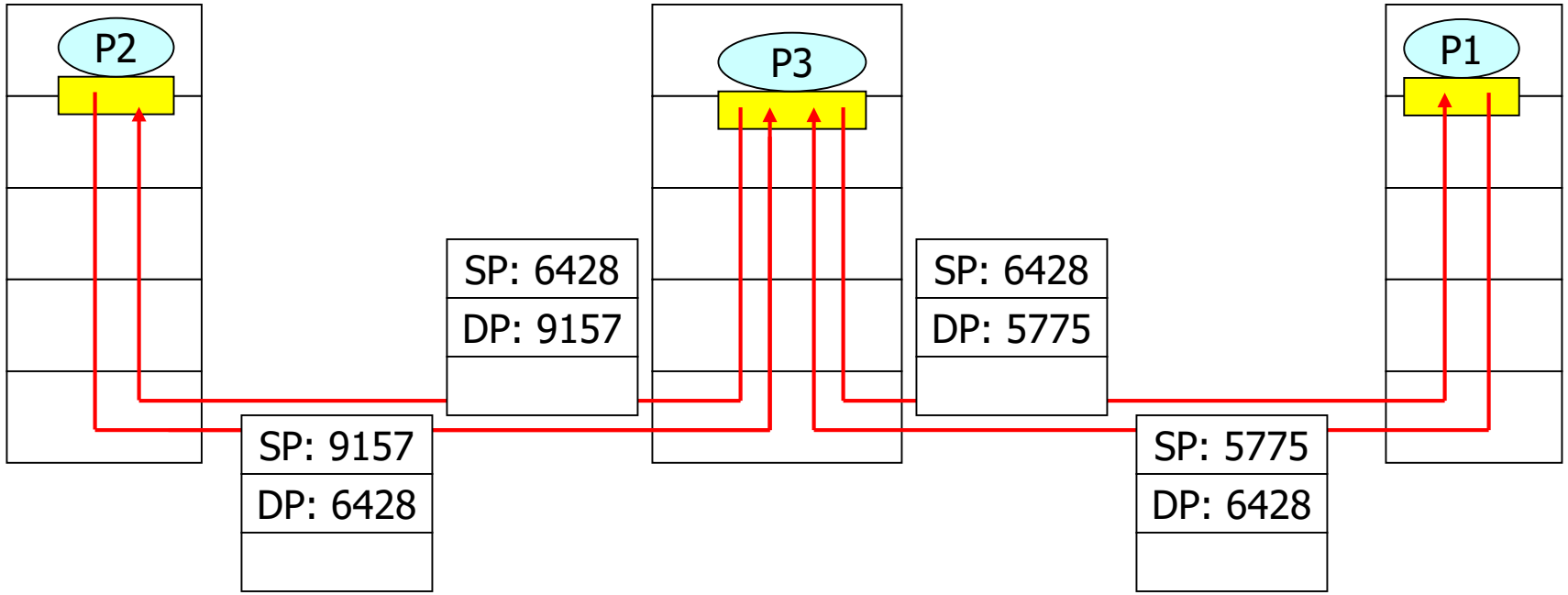
- Multiplexen: Zusammenführen der Segmente verschiedener Anwendungsprozesse durch Transportschicht auf dem Quellhost
  kombinieren von vielen anfragen verschiedener Anwendungen
- Demultiplexen: Ausliefern der Segmente an die verschiedenen Anwendungsprozesse durch Transportschicht des Zielhosts
  Wieder aufteilen auf die Zielanwendungen
- Anwendungsprozess vereinbart mit Transportschicht auf Quellhost Quellportnummer (wird entweder durch Anwendung gewählt oder ein freier Port wird vom Betriebssystem geliefert)
- realisiert z.B. durch Socket-API:

```
DatagramSocket serverSocket = new DatagramSocket(6428);
serverSocket.send(aPacket);
```

- UDP auf dem Zielhost erkennt an Zielportnummer (und nur daran), zu welcher Anwendung das Segment geliefert werden soll
  Quellport ist irrelevant (im gegensatz zu TCP)
- ein Anwendungsprozess kann mehrere Sockets besitzen

UDP

- Multiplexen und Demultiplexen, Beispiel:



Quelle: Kurose, Ross.
Computer Networking: A Top-Down Approach, 7th Ed.,
Pearson Education, 2017.

UDP

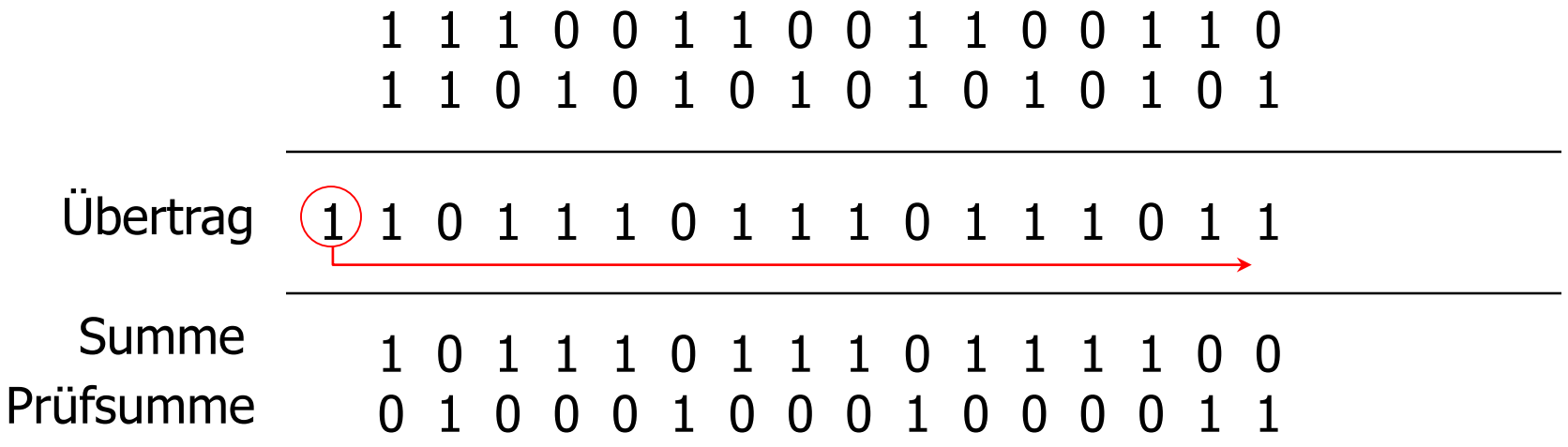
- Berechnung der Prüfsumme
  - Segment wird als Folge von Dualzahlen der Länge 16 Bit aufgefasst
  - diese werden in Einerkomplementarithmetik addiert
    - -x entsteht aus x durch Invertierung aller Bits
    - entsteht ein Übertrag, wird das Ergebnis inkrementiert
  - das Ergebnis wird invertiert, dies ist die Prüfsumme
  - der Sender berechnet die Prüfsumme und schreibt sie in das Segment
  - der Empfänger berechnet in gleicher Weise die Prüfsumme und addiert in Einerkomplementarithmetik die aus dem Segment gelesene Prüfsumme
  - falls kein Bitfehler vorliegt, ergibt sich als Ergebnis 1111111111111111_2, die Einerkomplement-Repräsentation von 0
  - einzelne Bitfehler werden erkannt, doppelte nicht
  - es gibt bessere Fehlererkennungsmechanismen

also in 16 bit slices aufteilen, alle addieren (nach einerk

besser Cyclic redundancy Check (basierend auf binären polynomen)

UDP

- Berechnung der Prüfsumme, Beispiel:



Ist das nicht einfach ein XOR

UDP

- Pseudo-Header
  - es ist in Wirklichkeit noch ein bisschen komplizierter ...
  - Pseudo-Header enthält Quell- und Ziel-IP-Adresse, Protokollnummer (17 für UDP) und Segmentlänge NOTHING MAKES SENSE ANYMORE. Eigentlich aus IP-protokol
  - UDP des Senders schreibt zunächst 0 in das Checksum-Feld, erstellt einen Pseudo-Header und berechnet die Prüfsumme zusammen für das UDP-Segment und den Pseudo-Header
  - diese Prüfsumme wird in das Checksum-Feld geschrieben Zuerst ausnullen, alles XOR-en
  - dann wird das Segment und der Pseudo-Header an IP weitergereicht
  - UDP des Empfängers erhält von IP das UDP-Segment und den Pseudo-Header, schreibt 0 in das Checksum-Feld und berechnet die Prüfsumme für Segment und Pseudo-Header
  - Vorteil: die Kontrolle der Prüfsumme erkennt auch Fehler in den IP-Adressen, z.B. fehlgeleitete Segmente
  - Nachteil: Verletzung des Schichtenprinzips (aber nur auf Endsystem)

UDP

Bitfehlerwahrscheinlichkeiten

- sei die Wahrscheinlichkeit eines einzelnen Bitfehlers p = 10^{-7}
- sei die Segmentlänge L = 10^4 Bits
- übliche vereinfachende Annahme (um überhaupt rechnen zu können): die Bitfehler der einzelnen Bits sind unabhängig voneinander

  Nicht realistisch, weil fehler gerne kaskadieren
- Wahrscheinlichkeit für mindestens einen Bitfehler im Segment:
  $1-(1-p)^L \approx 0{,}00099950 \approx 10^{-3}$

  ganzes segment hat keine Fehler
- Wahrscheinlichkeit für zwei Bitfehler im Segment:
  - Anzahl Paare: $\sum_{i=1}^{L-1} i = \frac{(L-1)\cdot L}{2} = \frac{(10^4-1)\cdot 10^4}{2} = \binom{10^4}{2} \approx \frac{10^8}{2}$

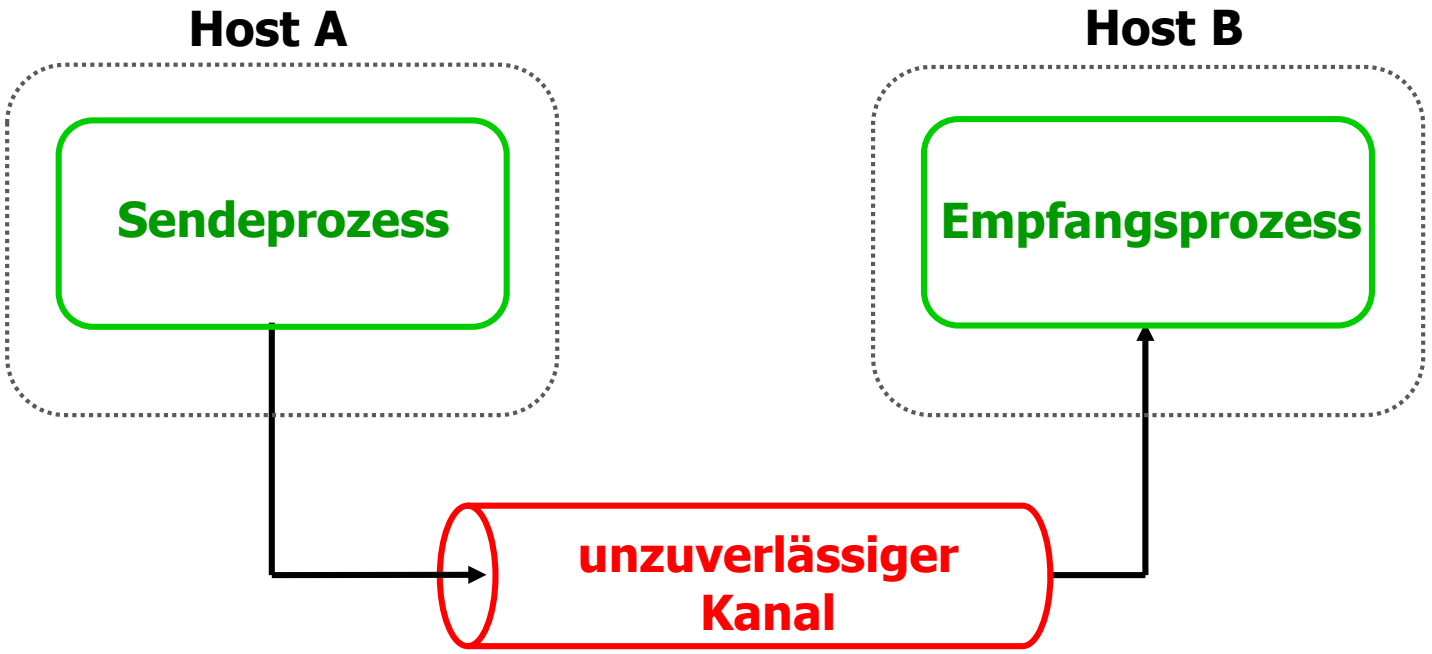
    Erster hat L-1 partner, zweiter L-2, dritter L-3,...
  - Wahrscheinlichkeit, dass ein bestimmtes Paar fehlerhaft ist: 10^{-14}
  - Wahrscheinlichkeit, dass ein beliebiges Paar fehlerhaft ist: $10^8 \cdot 10^{-14}/2 = 10^{-6}/2$
- wie lange dauert es im Mittel, bis ein Segment mit zwei Bitfehlern auftritt bei
  a) 10 Mbps und
  b) 10 Gbps?

#segmente/s = 10Mbps/10^4bits = 1000/sfehlerrate/s = #segmente/s *10^(-6)/2bei

Transportschicht

- Einführung
- UDP
- Fehlerkontrolle
  - Stop-and-Wait
  - Go-Back-N und Selective Repeat
  - Leistungsanalyse
- TCP
- TLS
- QUIC

Fehlerkontrolle

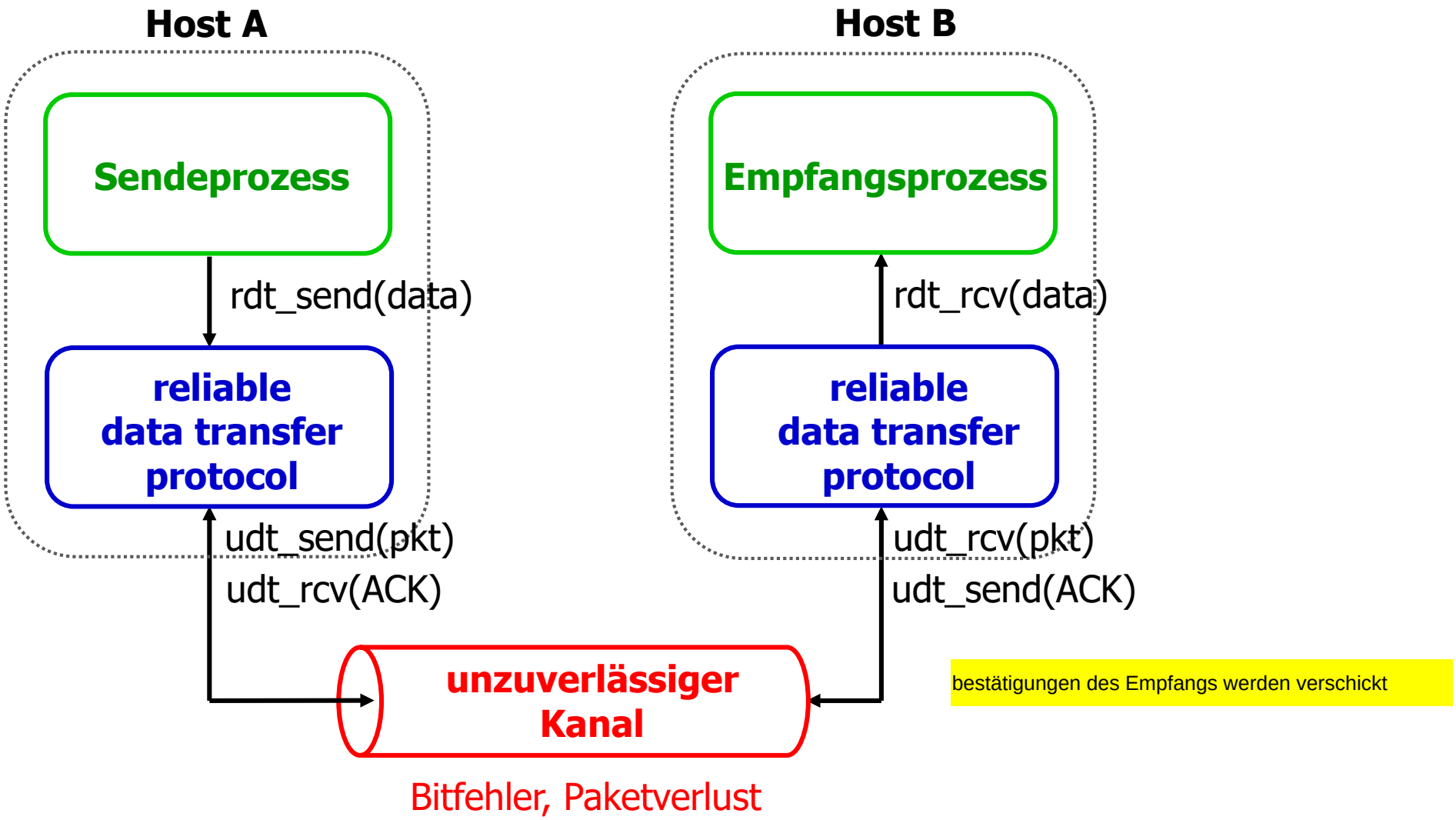


- Rauschen, Pufferüberläufe, Ausfälle von Komponenten verursachen Bitfehler und Paketverluste
- kann durch Protokoll mit Fehlererkennung, Bestätigungen und Sendewiederholungen ausgeglichen werden

Fehlererkennung kann aber o.b.d.A. nicht bei Pakerverlust abhilfe schaffen

Rückwärtsfehlerkontrolle oder automatic repeat request

Fehlerkontrolle



Fehlerkontrolle

- 3 grundlegende Protokolle für zuverlässigen Transport
  - Stop-and-Wait
    - Sender fügt zur Fehlererkennung Prüfsumme oder besser Cyclic Redundancy Check (CRC) zu
    - Empfänger schickt Bestätigung (acknowledgment, ACK)
    - wenn diese nach einem Timeout nicht eintrifft, wird das Paket erneut gesendet
    - dadurch können evtl. Duplikate gesendet werden, um diese zu erkennen, benötigt man noch Sequenznummern (SQN)
    - bei großem Bitraten-Verzögerungsprodukt: Sender ist die meiste Zeit blockiert, ineffizient
  - Schiebefensterprotokolle (sliding window protocols)
    - mehrere Pakete auf einmal senden, um Kanal zu füllen
    - Go-Back-N und Selective Repeat
    - unterscheiden sich bei Timeout, Bestätigungen, Sendewiederholung

Die Leitung hat pufferwirkung, aber wenn man nicht kontinuierlich sendet, ist die Leitung immer unterbelegt A=R*l/v. Kein Pip

Transportschicht

- Einführung
- UDP
- Fehlerkontrolle
  - Stop-and-Wait
  - Go-Back-N und Selective Repeat
  - Leistungsanalyse
- TCP
- TLS
- QUIC

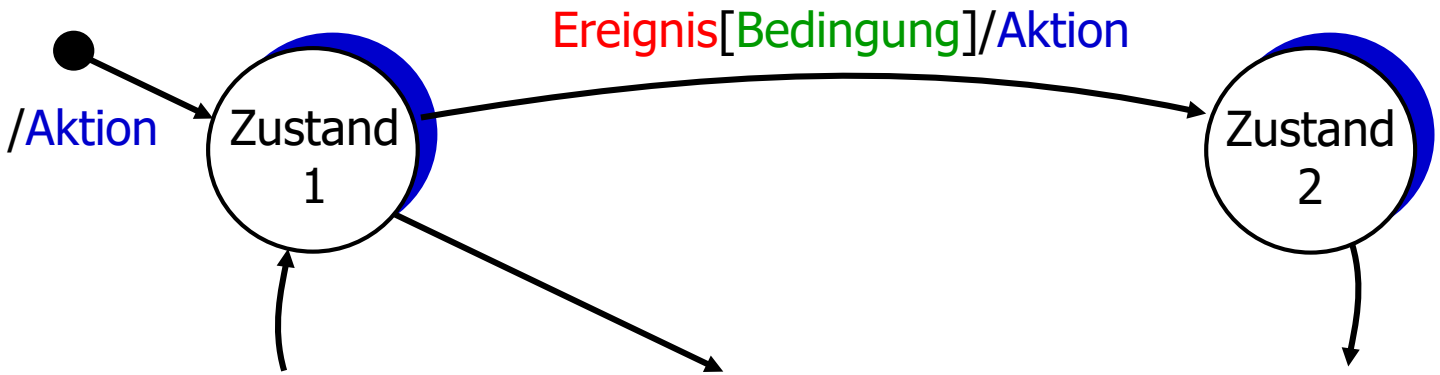
Stop-and-Wait

- wie funktioniert nun Stop-and-Wait genau?
- zunächst eine informelle Beschreibung:

  - Verhalten des Senders
    1. sende Paket mit aktueller SQN und starte Timer
    2. wenn ein ACK ohne Bitfehler und mit aktueller SQN vor Ablauf des Timeouts zurückkommt, inkrementiere SQN und gehe zu 1
    3. wenn der Timeout abläuft, sende das Paket erneut, starte den Timer erneut und gehe zu 2
  - Verhalten des Empfängers
    - wenn Paket ohne Bitfehler und mit aktueller SQN ankommt, sende ACK mit aktueller SQN und inkrementiere SQN, sonst sende das letzte ACK erneut

Stop-and-Wait

- Beschreibung durch UML-Statecharts
  - ein Statechart befindet sich immer in einem Zustand, der schwarze Punkt kennzeichnet den initialen Zustand
  - ein Zustandsübergang findet statt, wenn das Ereignis ausgelöst wurde und die Bedingung erfüllt ist
  - wenn ein Zustandsübergang stattfindet, wird die Aktion durchgeführt
  - zur Steigerung der Flexibilität gibt es auch Variablen

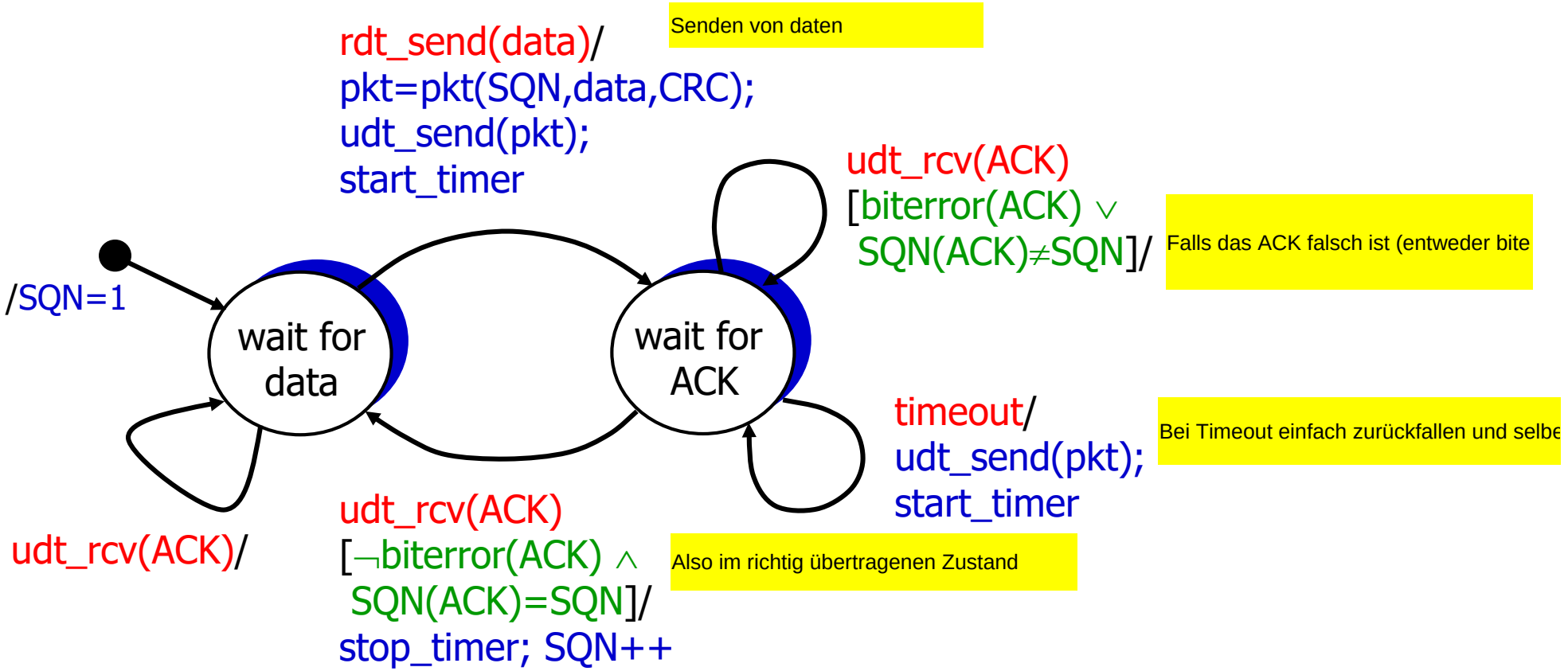


Stop-and-Wait

- Bemerkungen zu den Statecharts
  - Statecharts stellen eine Variante von endlichen Automaten dar
  - Ereignisse, Bedingungen und Aktionen werden oft durch Pseudocode beschrieben, man erhält eine halbformale Beschreibung
  - das Verhalten von Protokollen wird oft durch solche oder ähnliche Automaten dargestellt
  - es gibt auch Werkzeuge, die dies unterstützen: Protokolle können so spezifiziert werden und daraus der Code generiert werden sowie Analysen, Simulationen und Tests durchgeführt werden
  - man kann daraus gut Implementierungen ableiten: eine große Fallunterscheidung für die möglichen Ereignisse in den verschiedenen Zuständen
  - hier werden Statecharts einfach zur genauen Darstellung des Stop-and-Wait-Protokolls und später von weiteren Protokollen verwendet
  - die Darstellung ist durch Kurose/Ross motiviert, unterscheidet sich aber von den Automaten in dem Buch

Stop-and-Wait

- Sender:

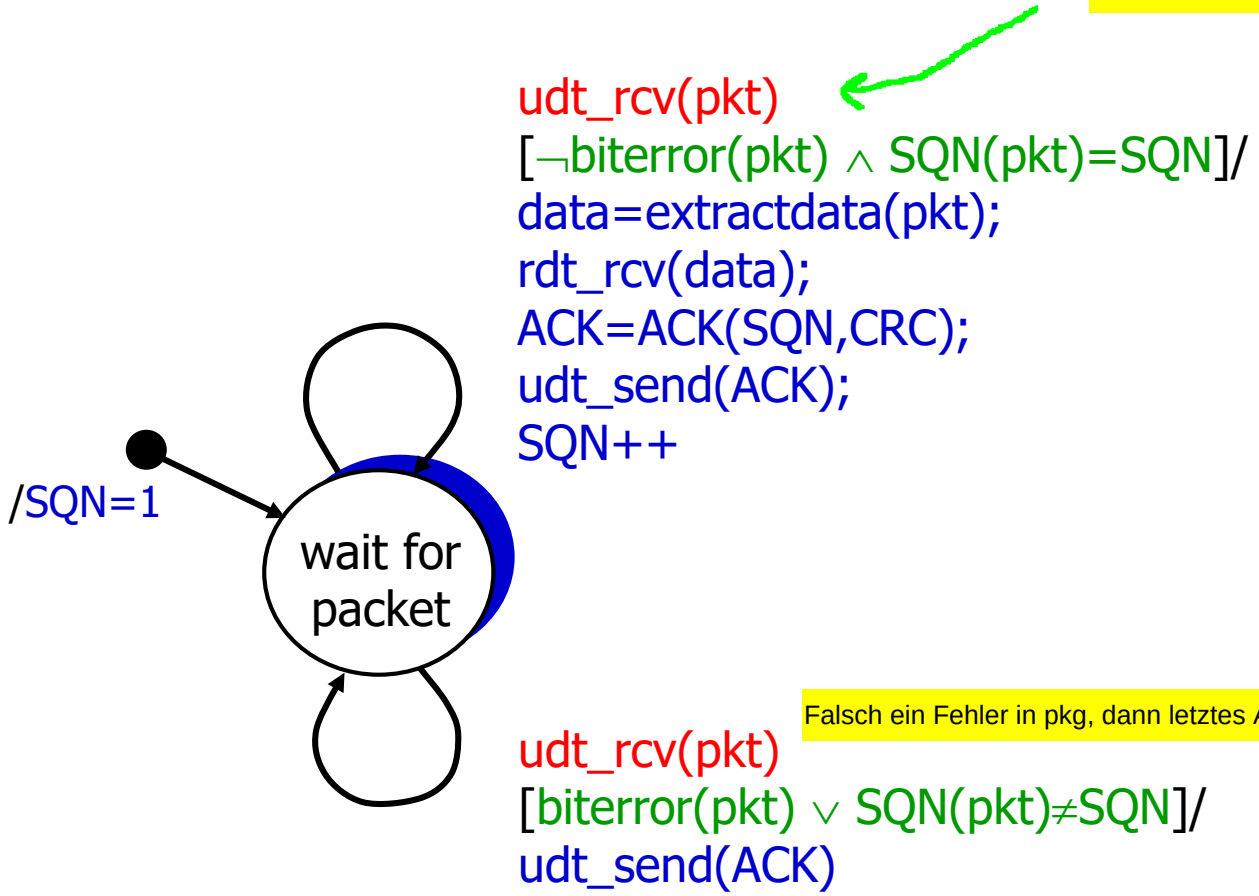


nach: Kurose, Ross. *Computer Networking: A Top-Down Approach*, 7th Ed., Pearson Education, 2017.

Stop-and-Wait

- Empfänger:

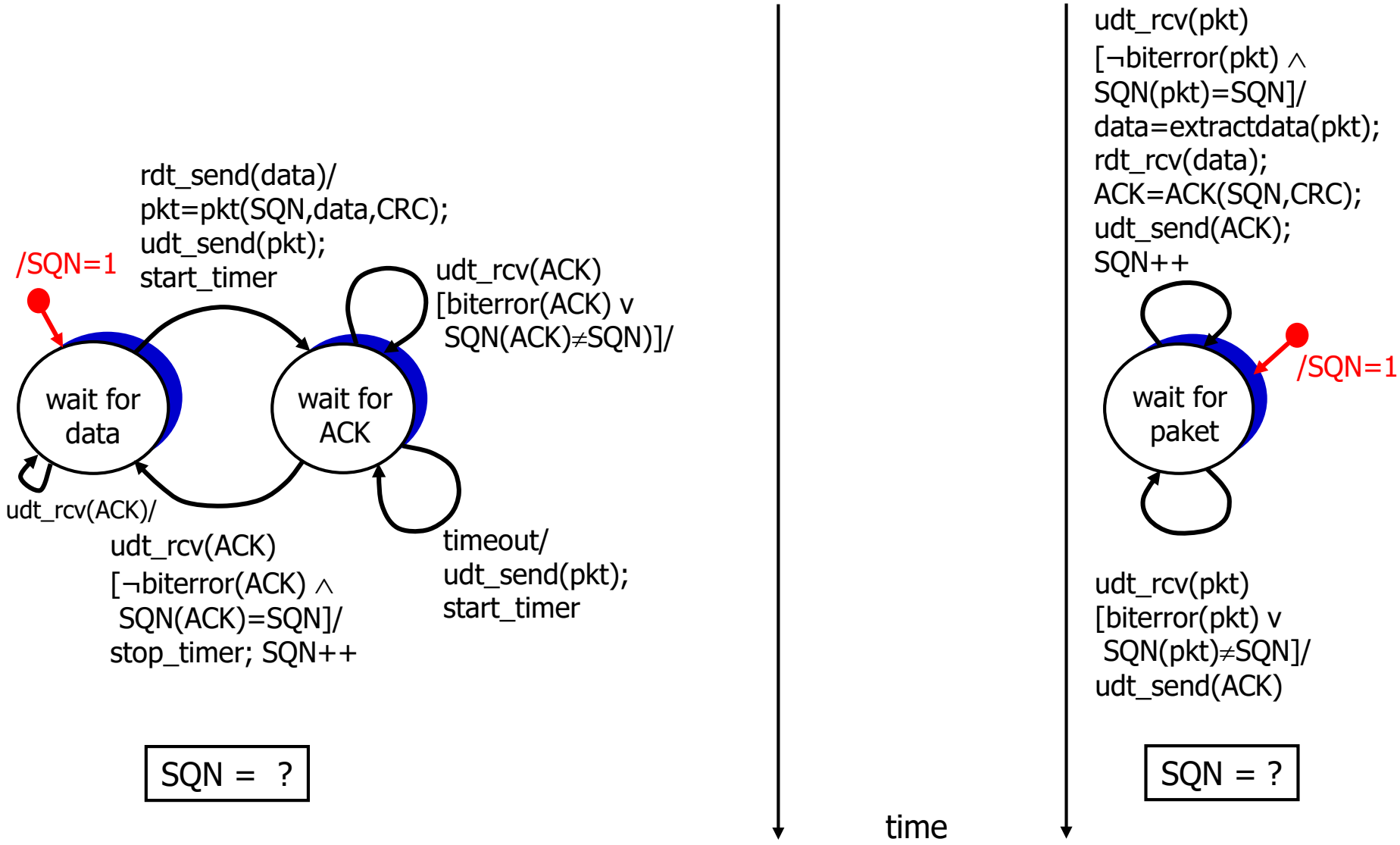
empfangen, prüfen.Wenn richtig dann aufs nächste package warten und AC



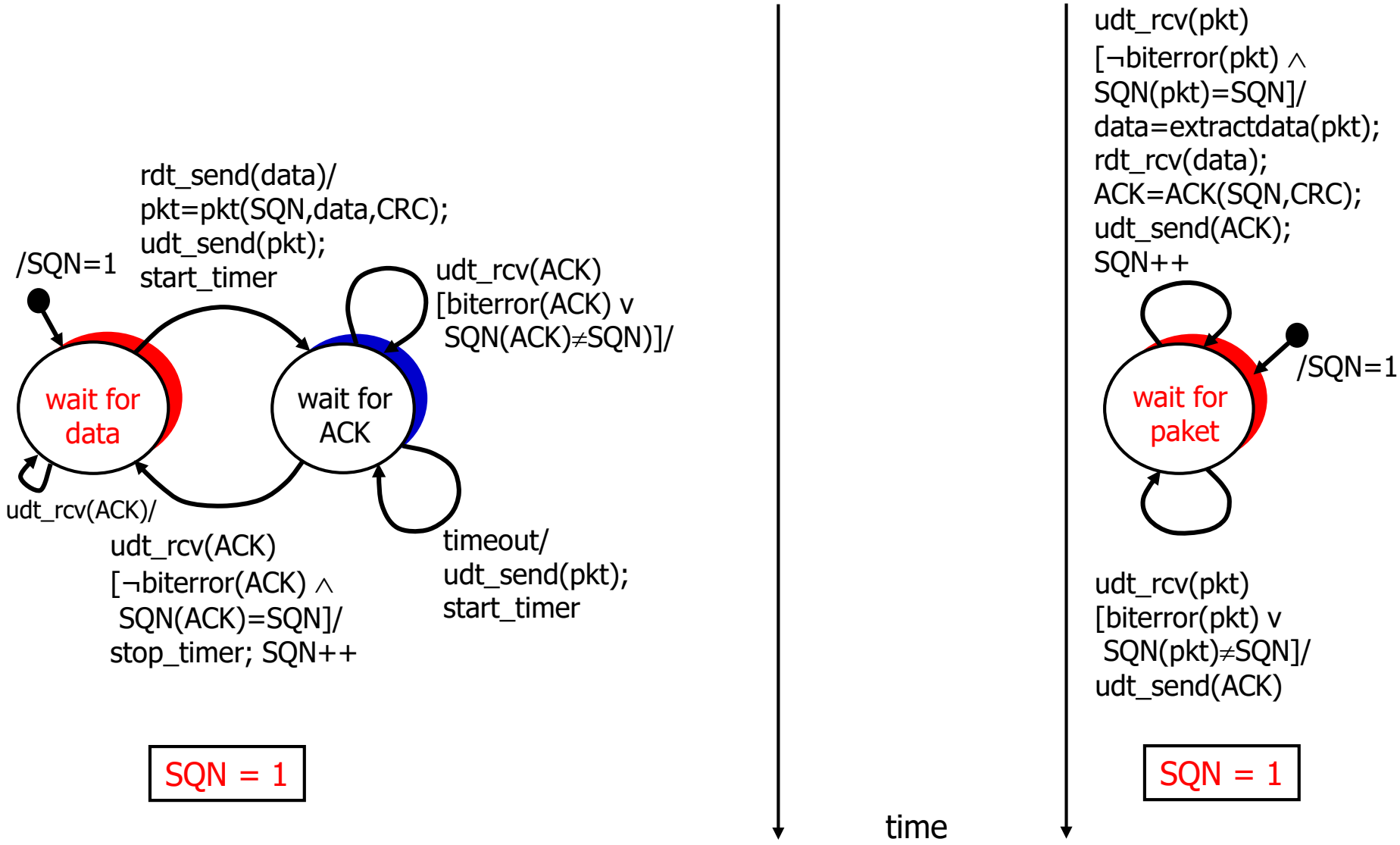
Falsch ein Fehler in pkg, dann letztes ACK nochmal senden (implizites fehlersignal)

nach: Kurose, Ross. *Computer Networking: A Top-Down Approach*, 7th Ed., Pearson Education, 2017.

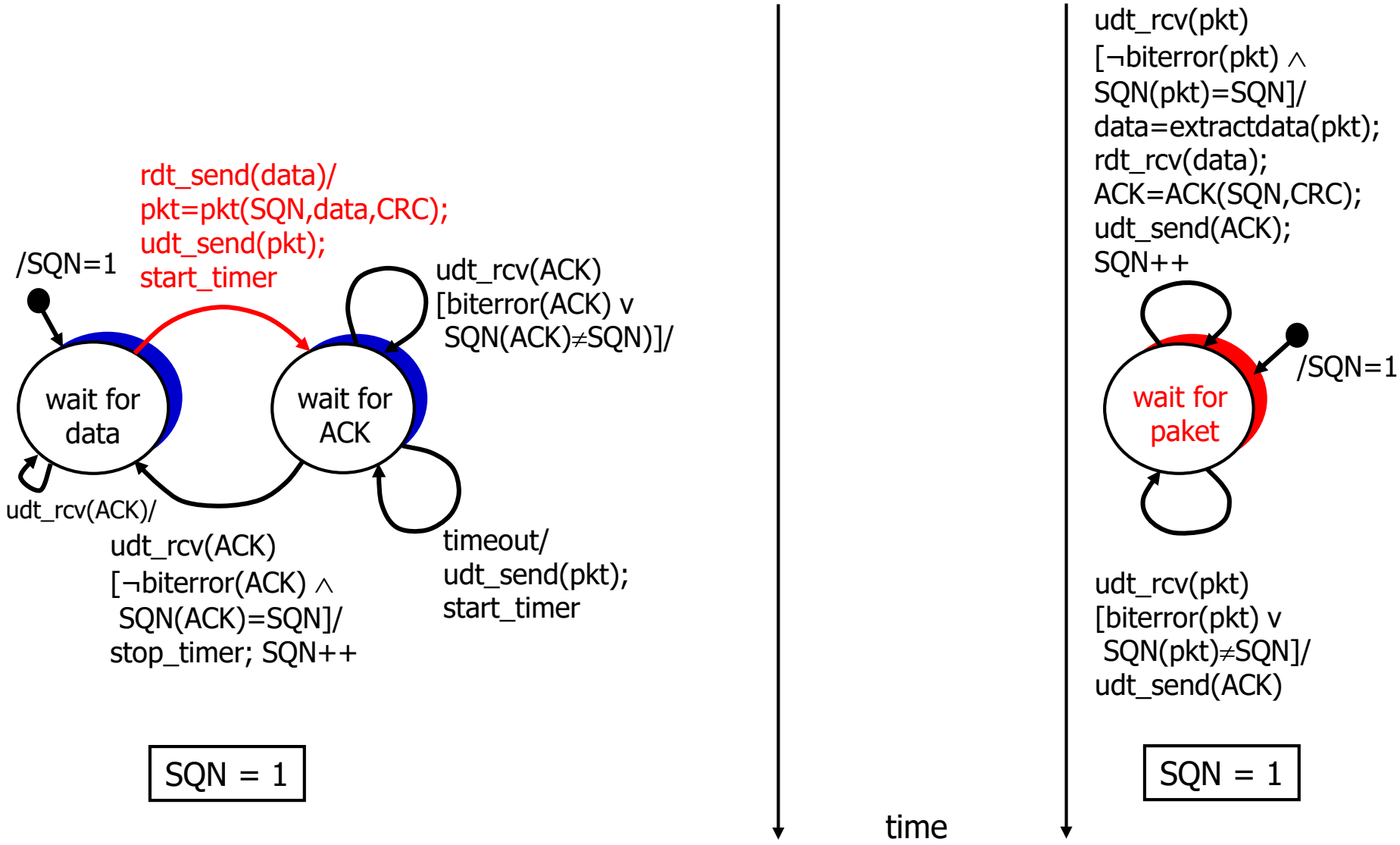
Stop-and-Wait: normaler Ablauf



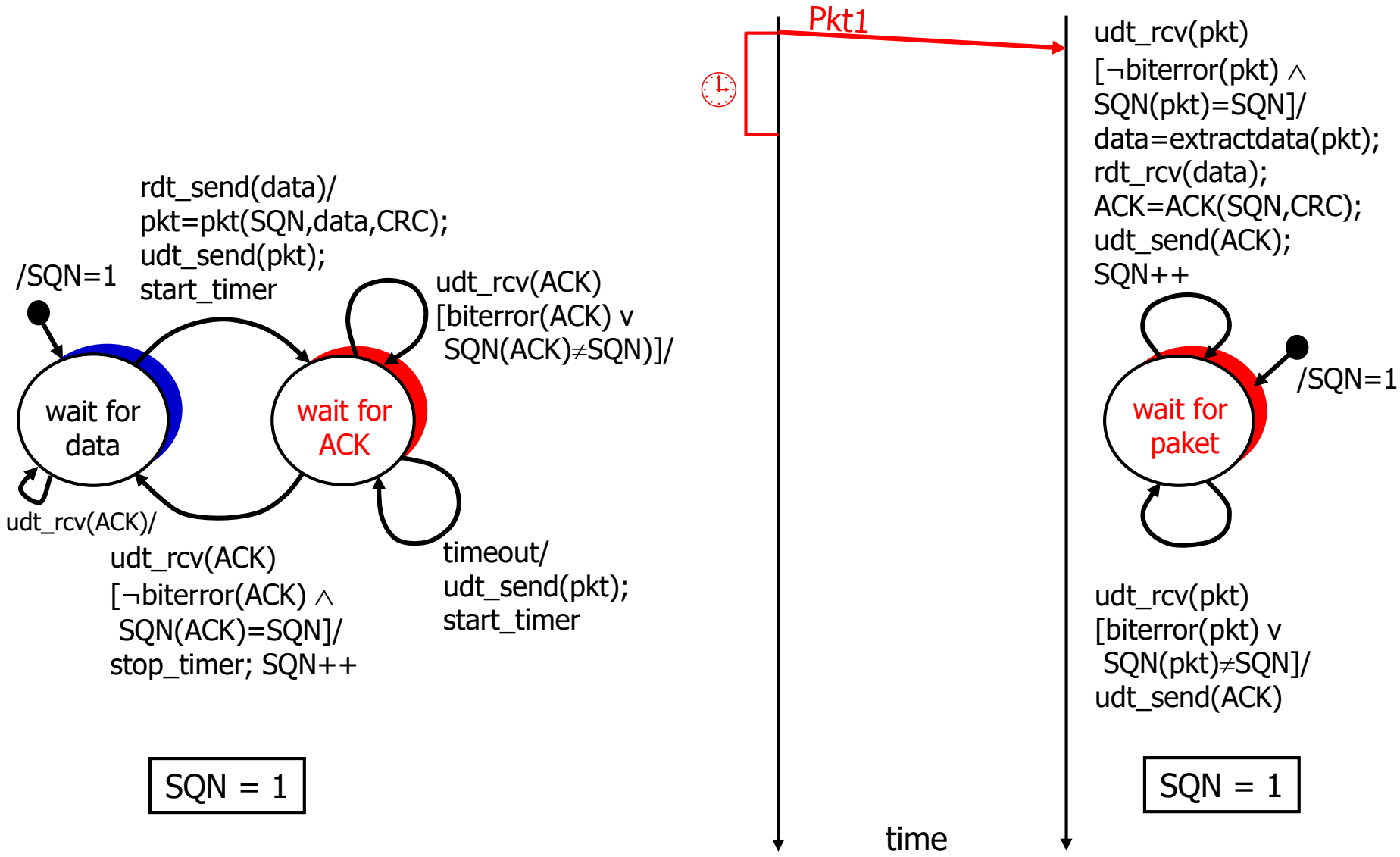
Stop-and-Wait: normaler Ablauf



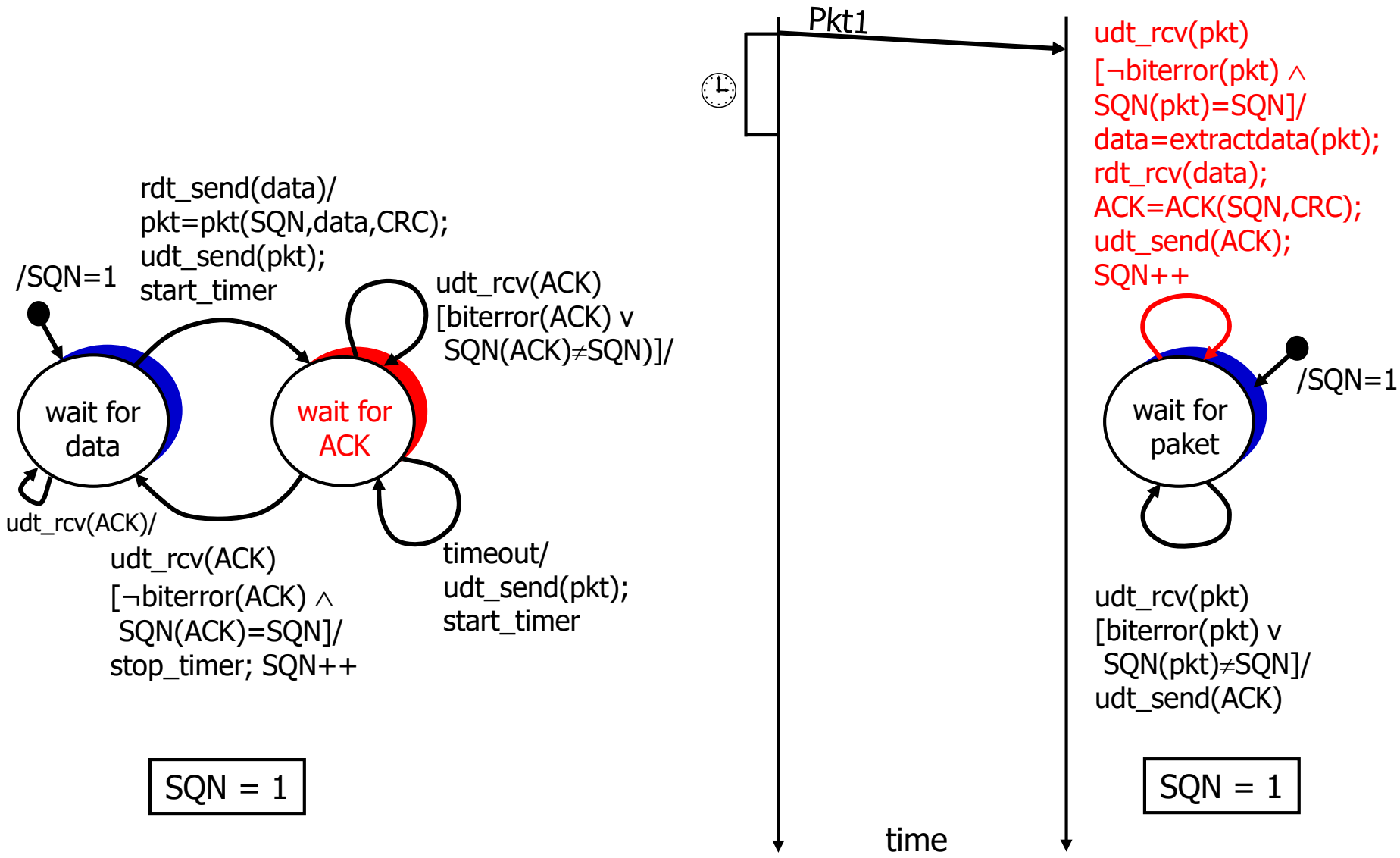
Stop-and-Wait: normaler Ablauf



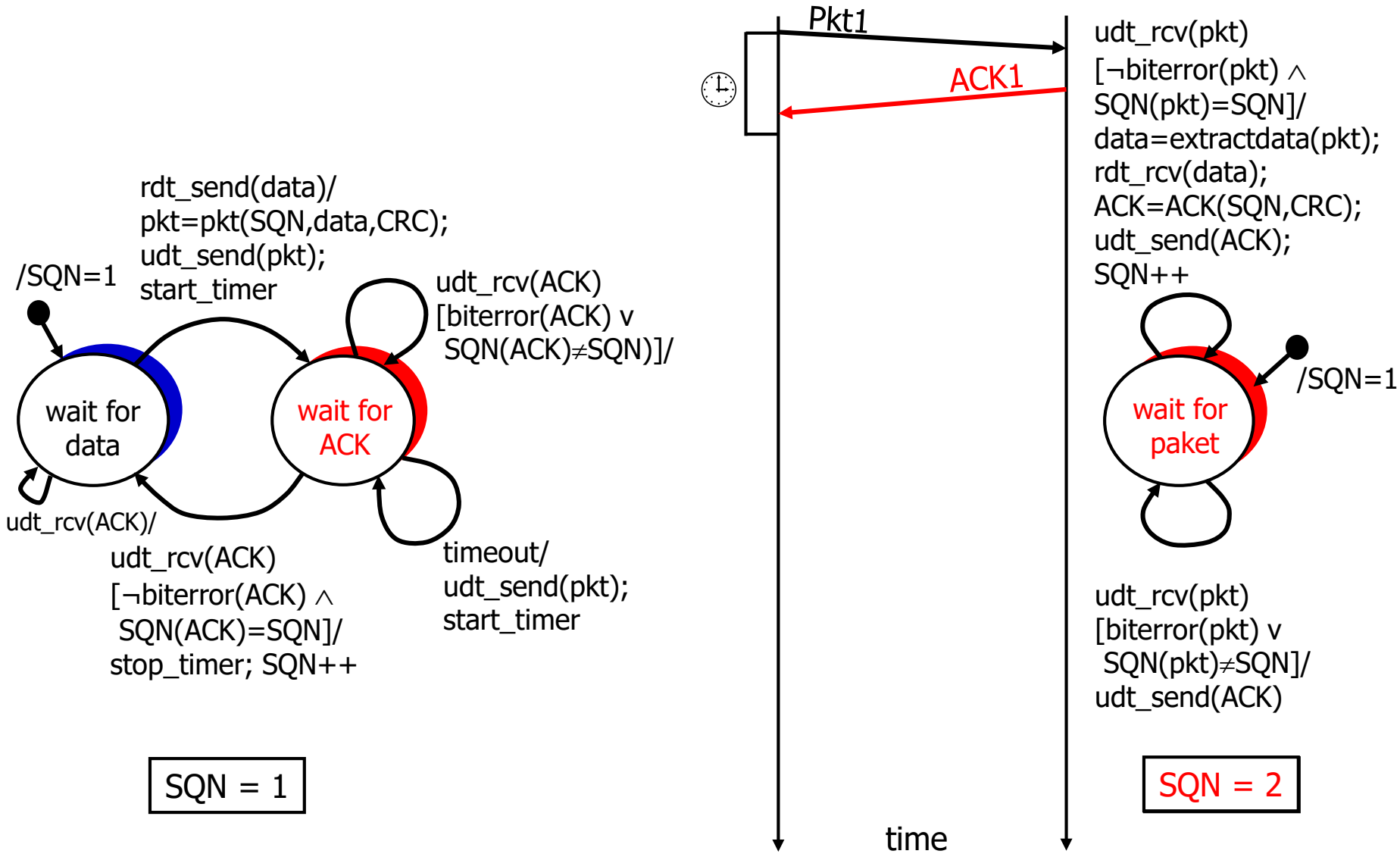
Stop-and-Wait: normaler Ablauf



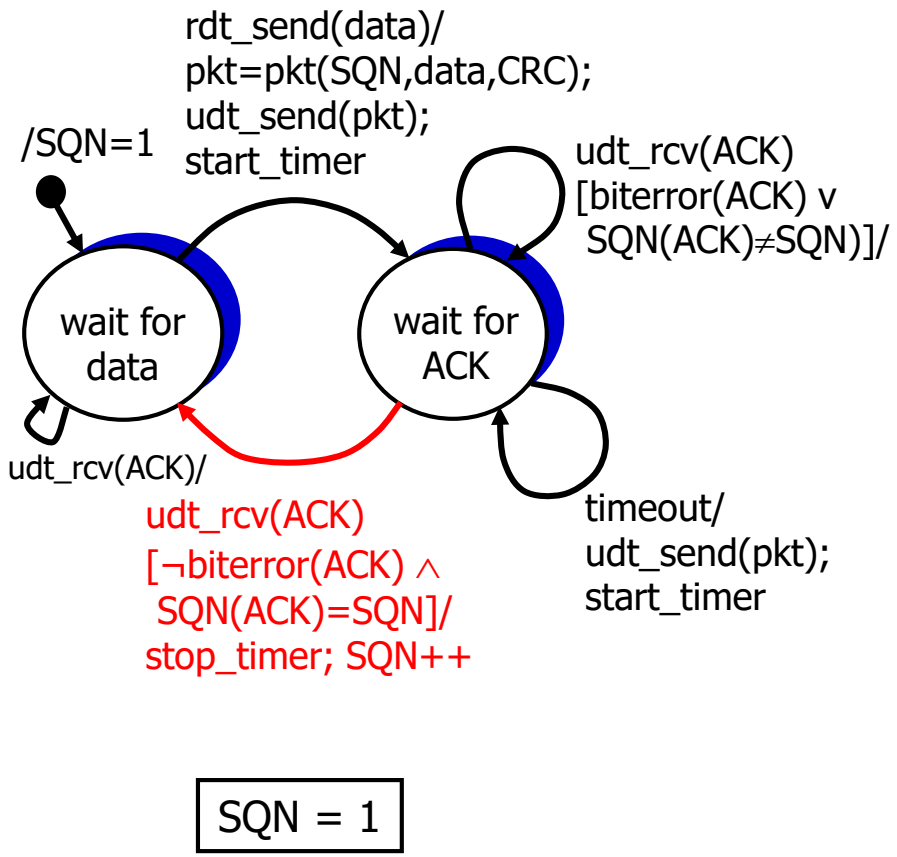
Stop-and-Wait: normaler Ablauf

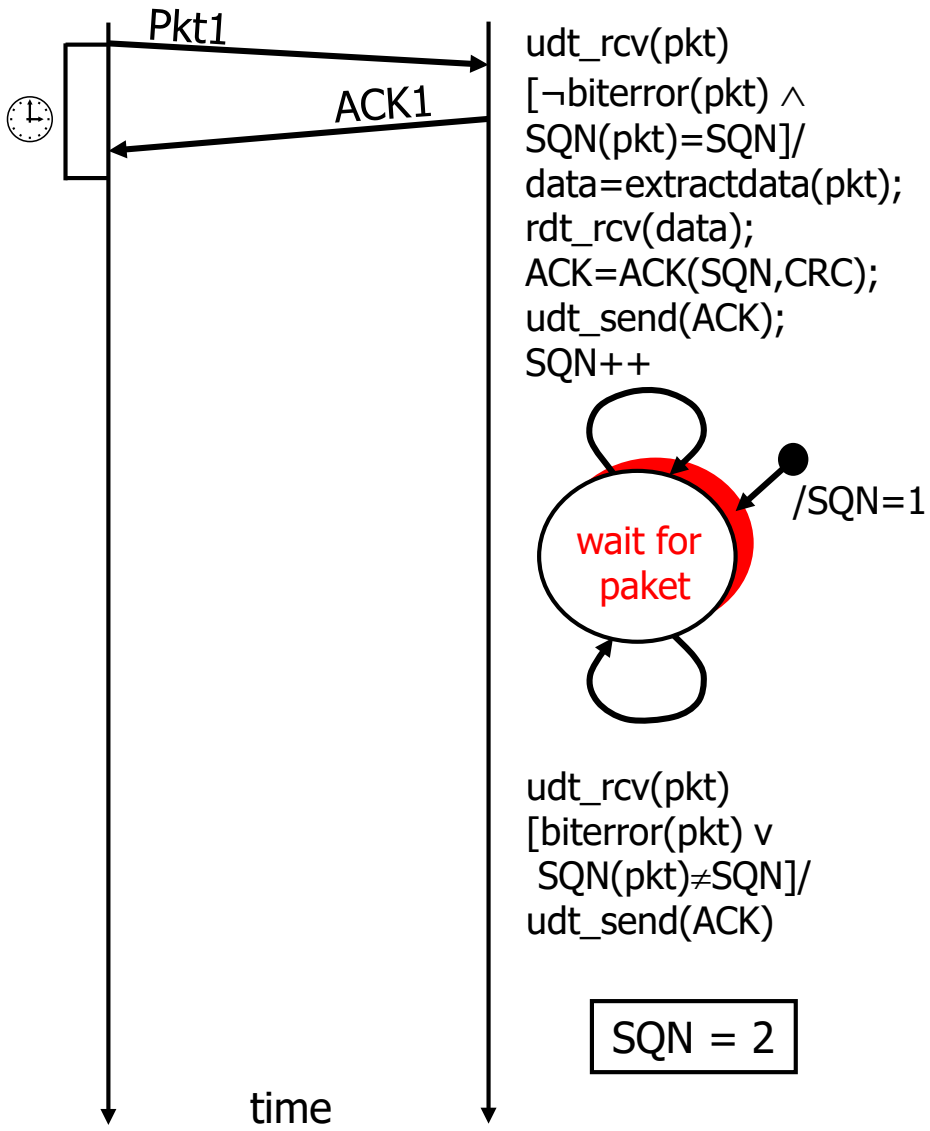


Stop-and-Wait: normaler Ablauf

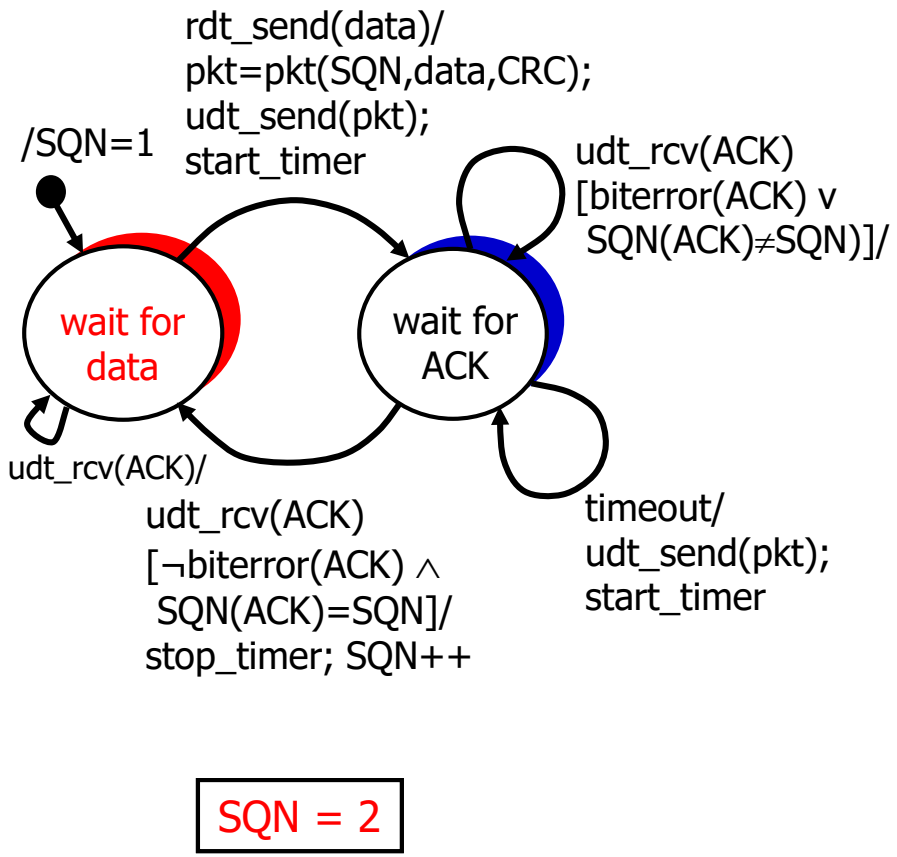


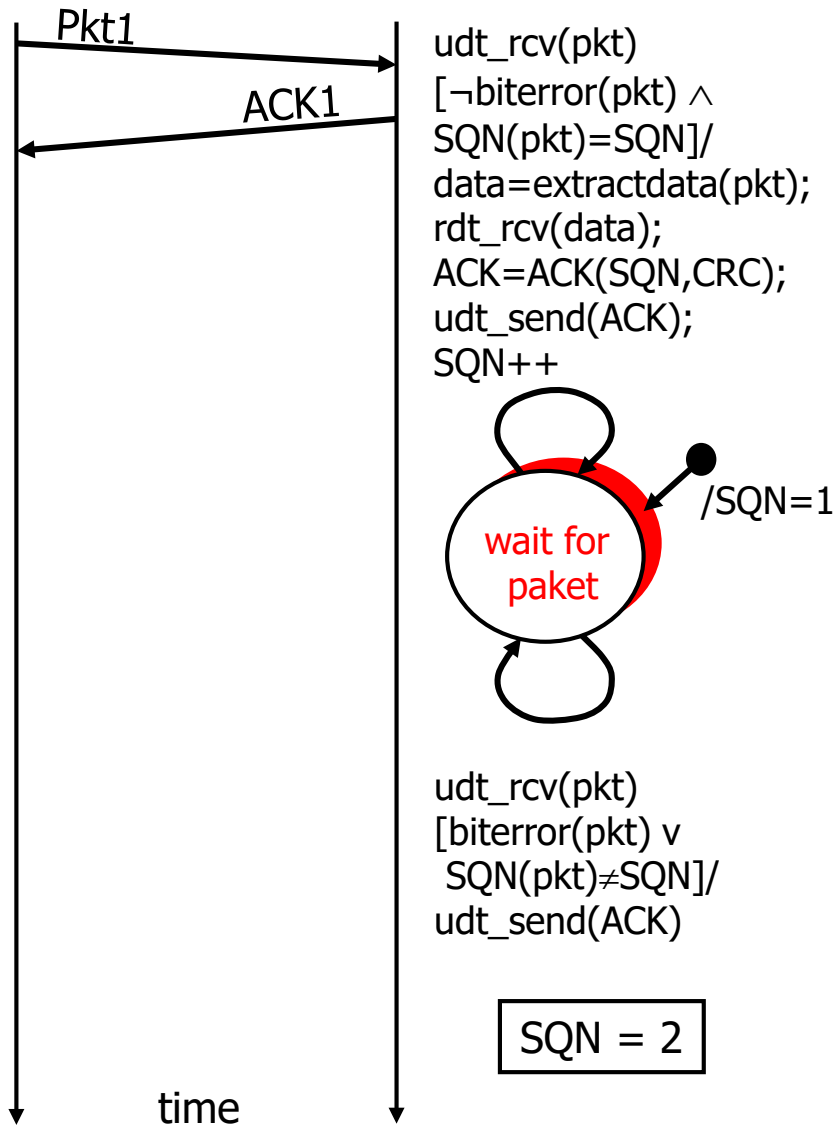
Stop-and-Wait: normaler Ablauf



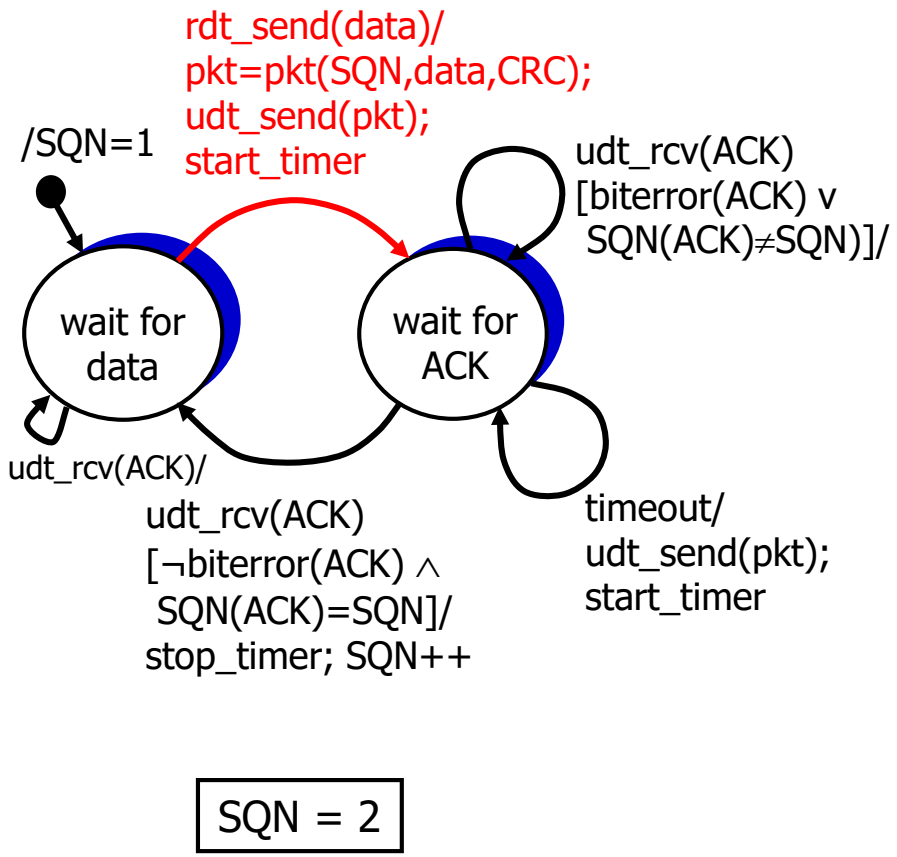


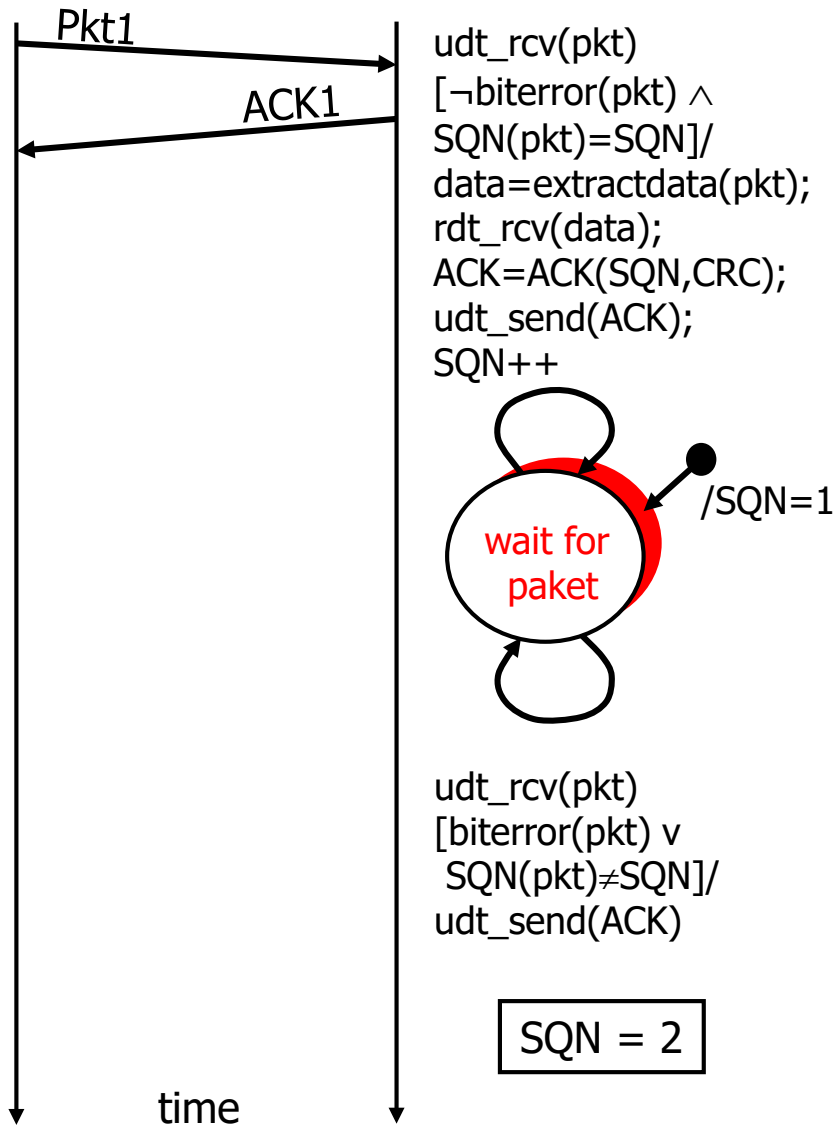
Stop-and-Wait: normaler Ablauf



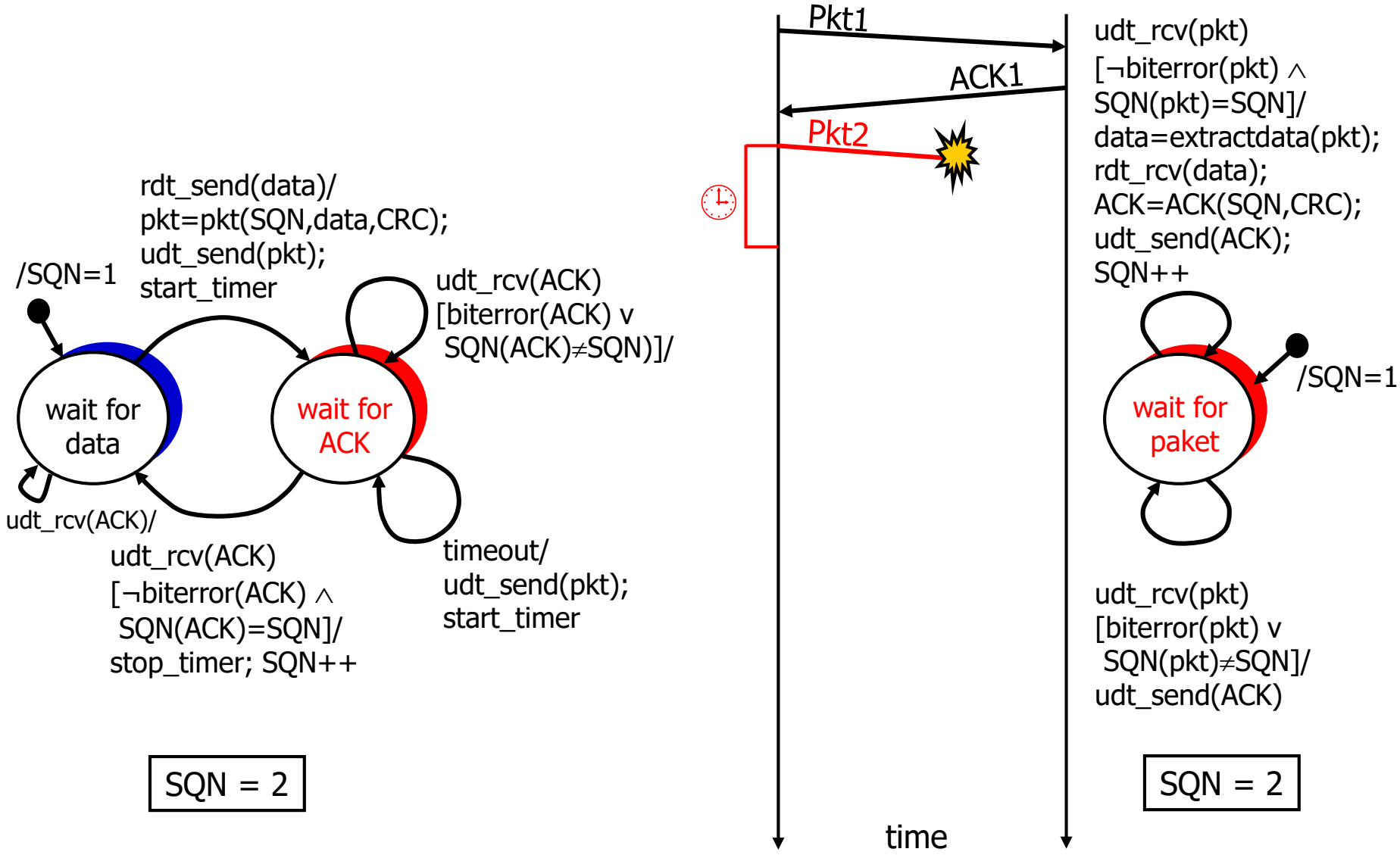


Stop-and-Wait: Paketverlust

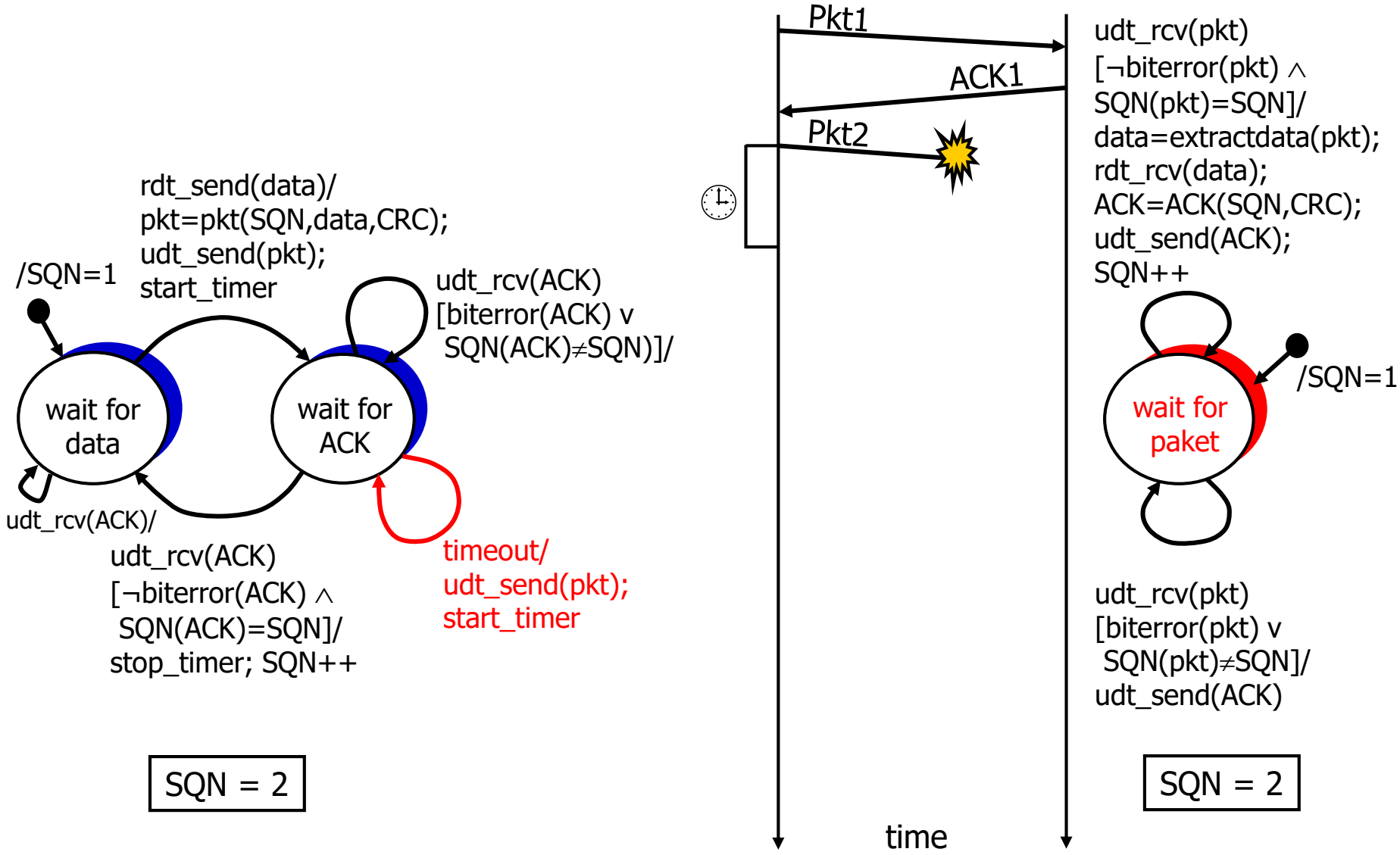




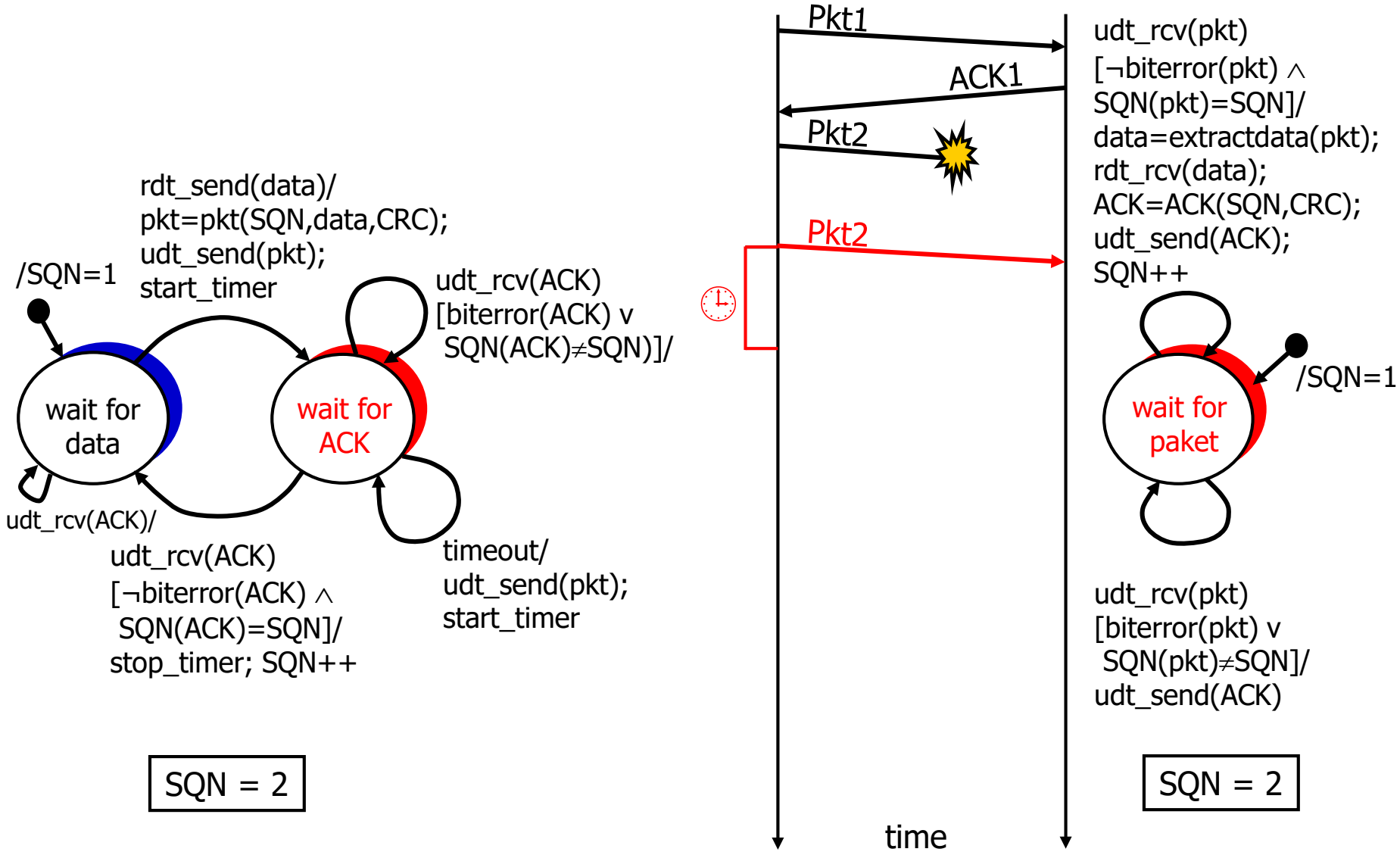
Stop-and-Wait: Paketverlust



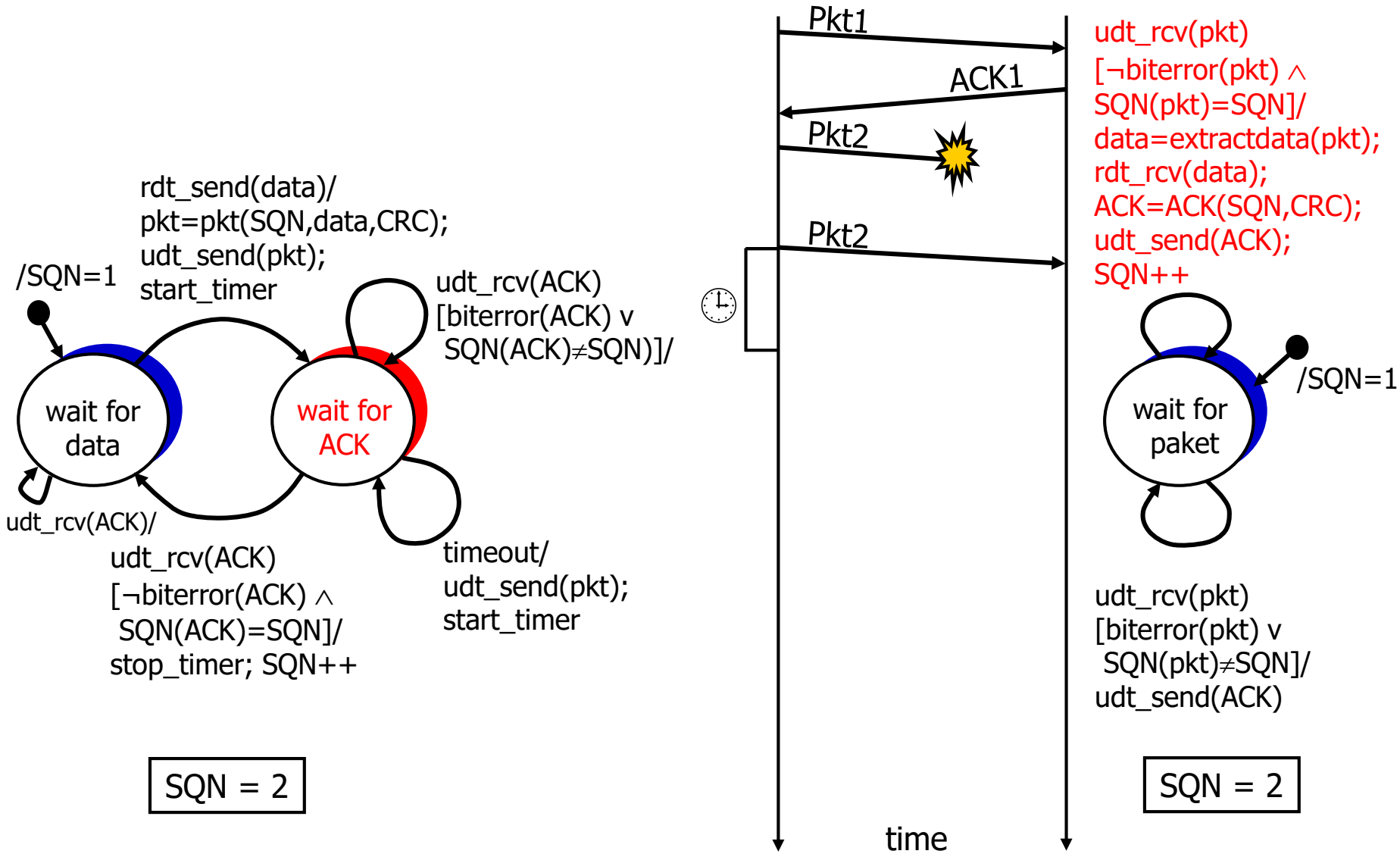
Stop-and-Wait: Paketverlust



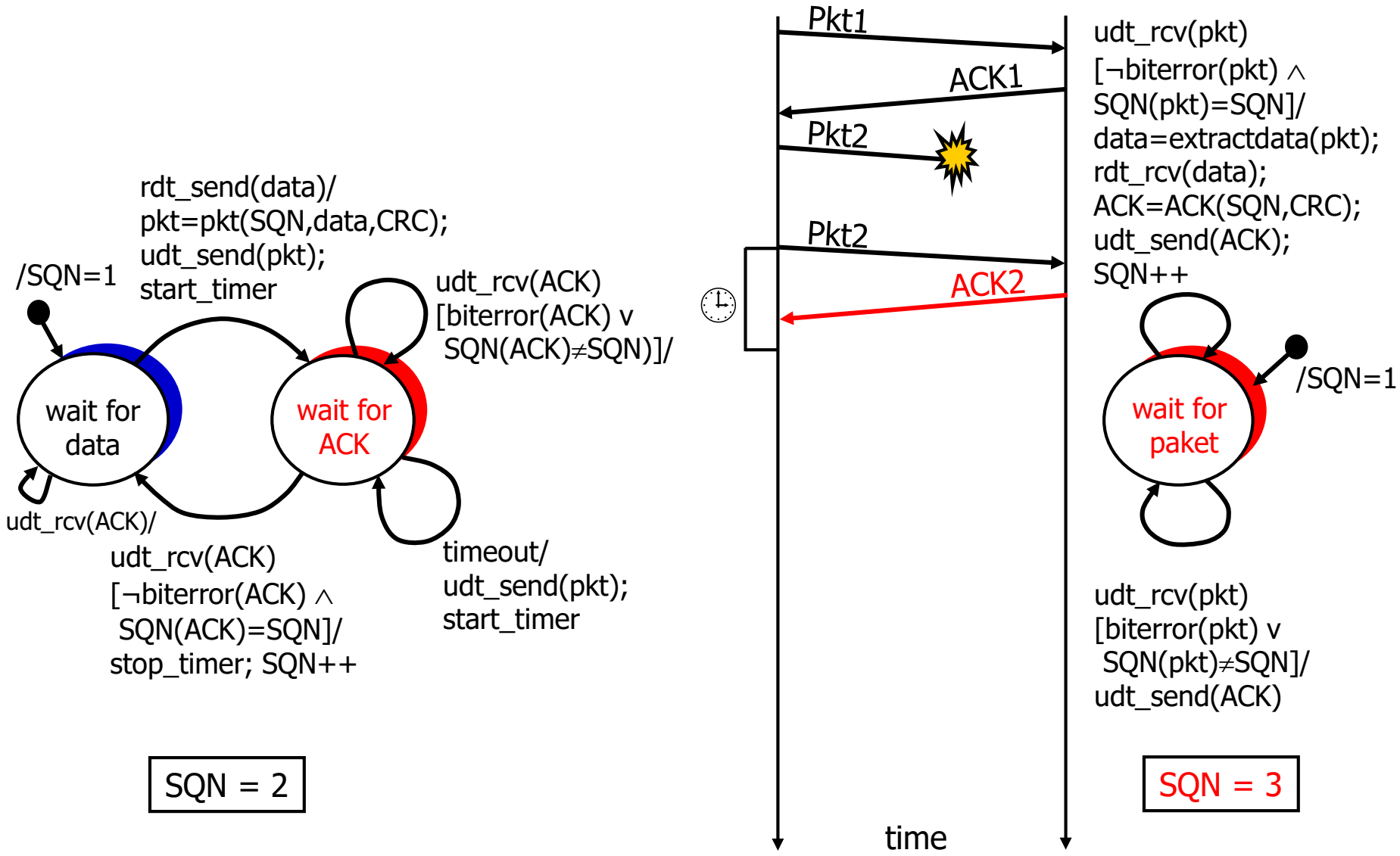
Stop-and-Wait: Paketverlust



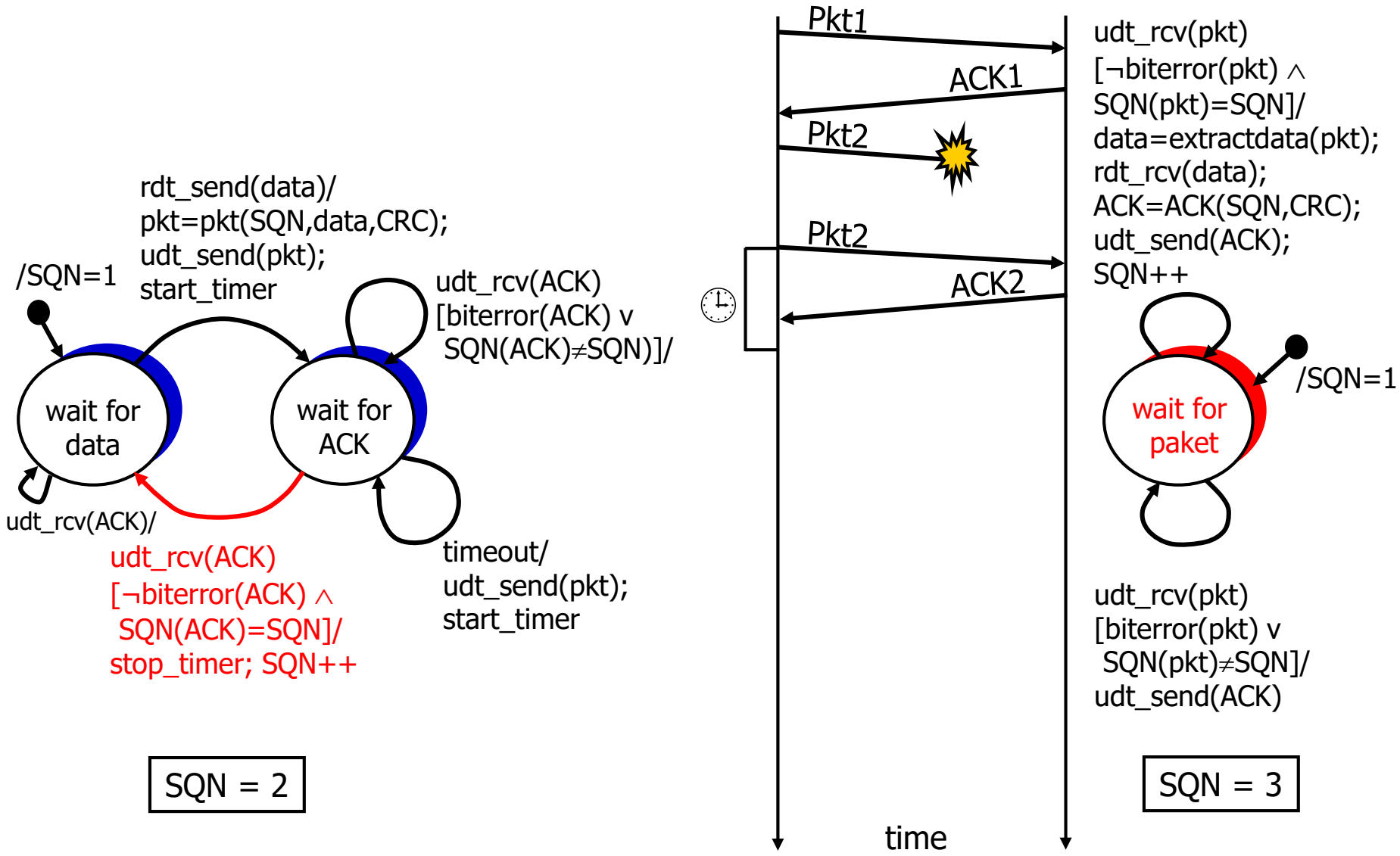
Stop-and-Wait: Paketverlust



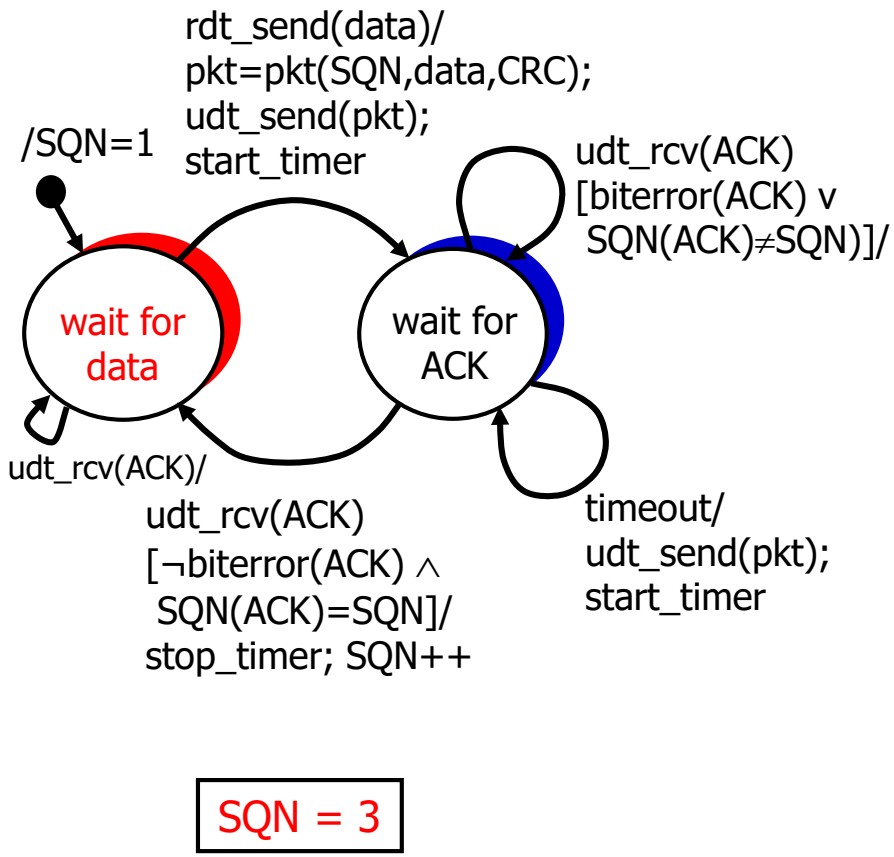
Stop-and-Wait: Paketverlust

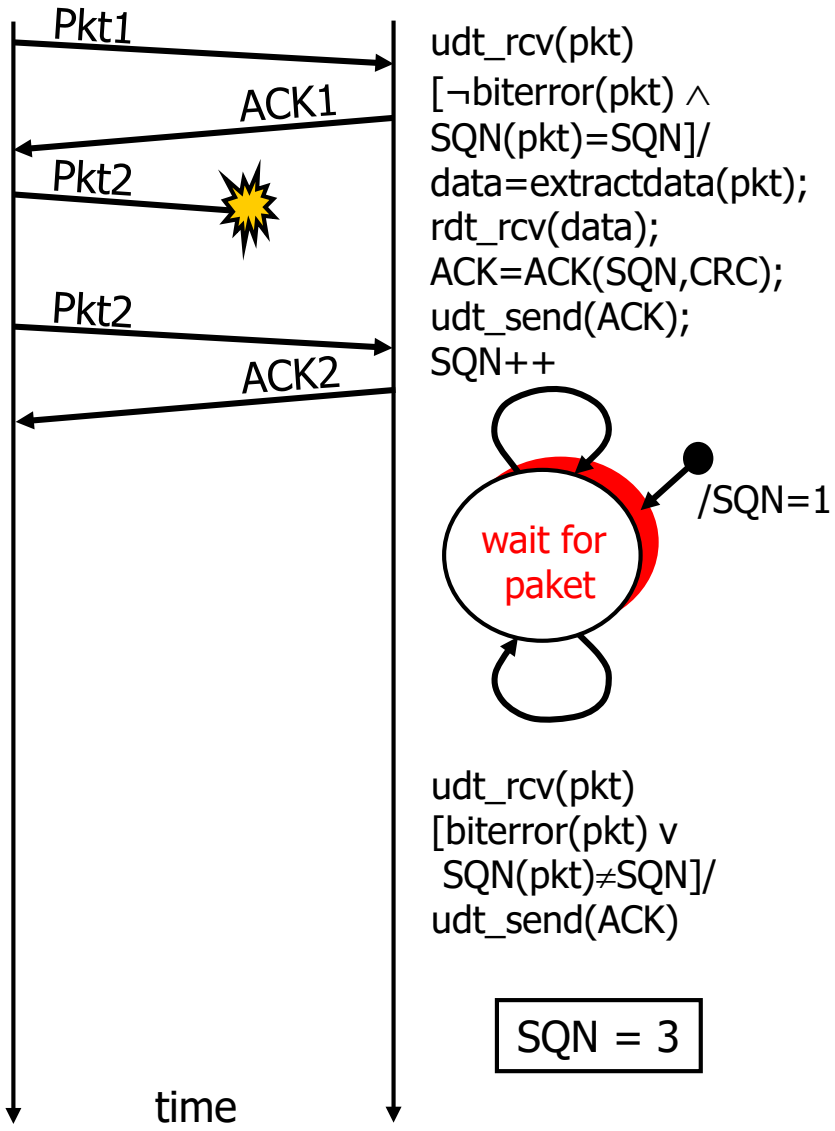


Stop-and-Wait: Paketverlust

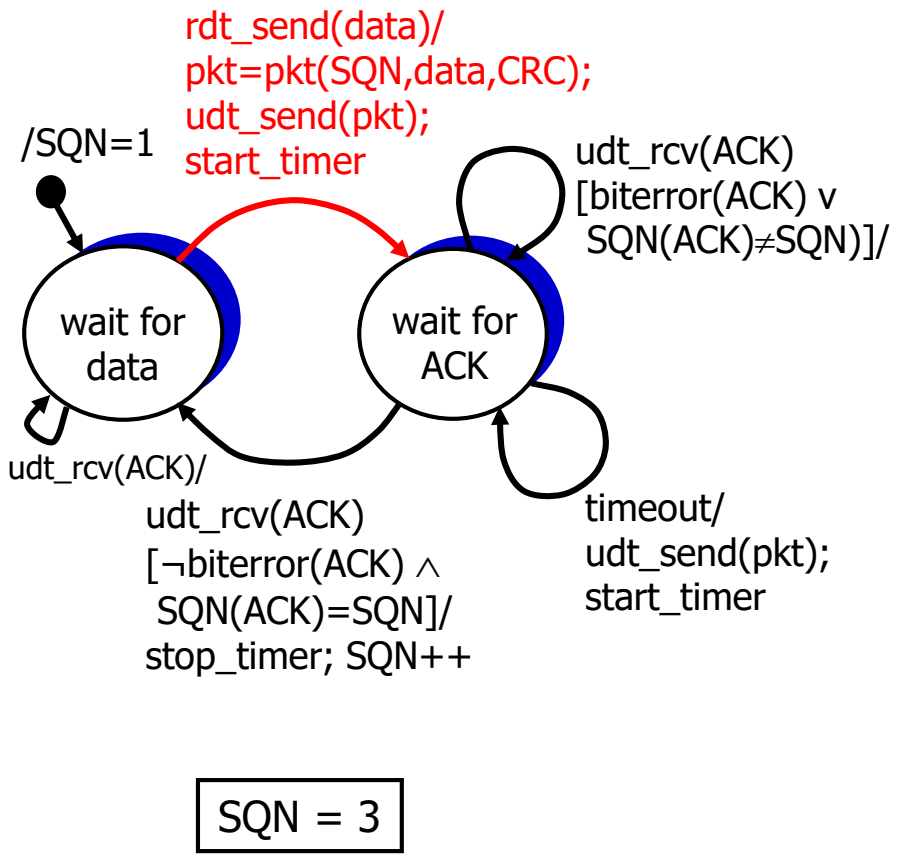


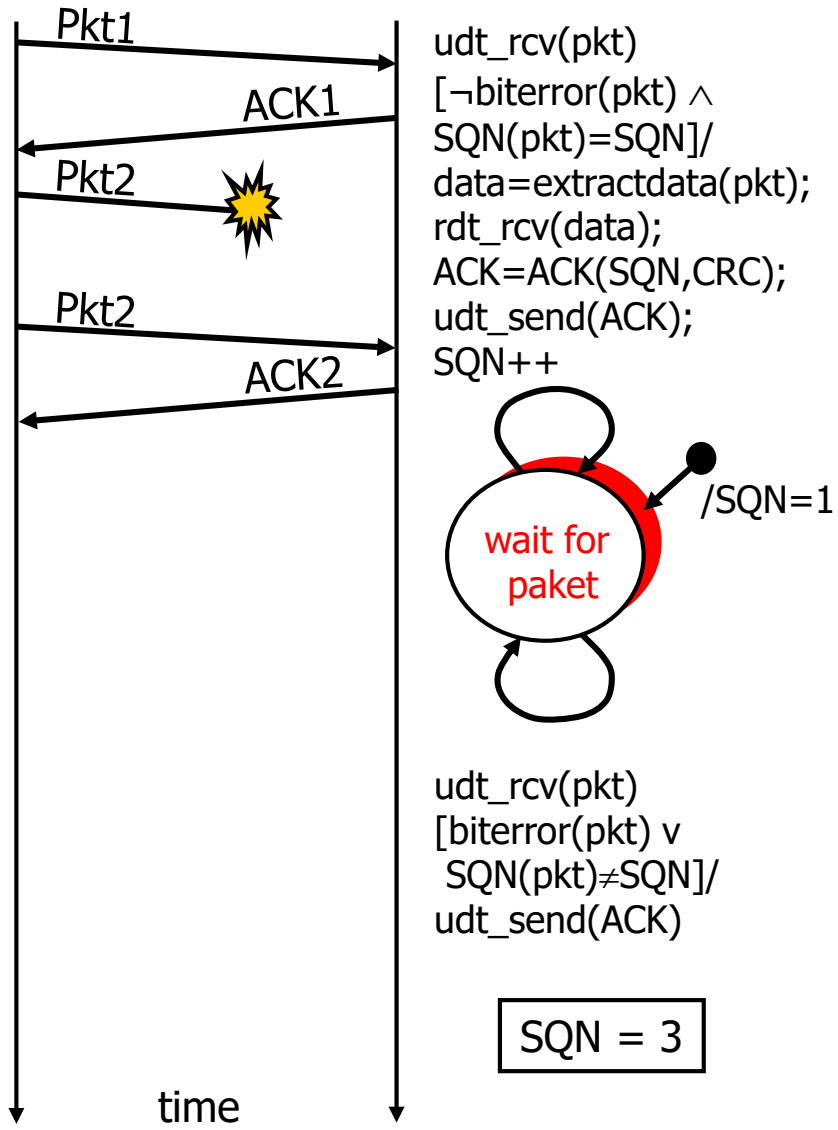
Stop-and-Wait: Paketverlust



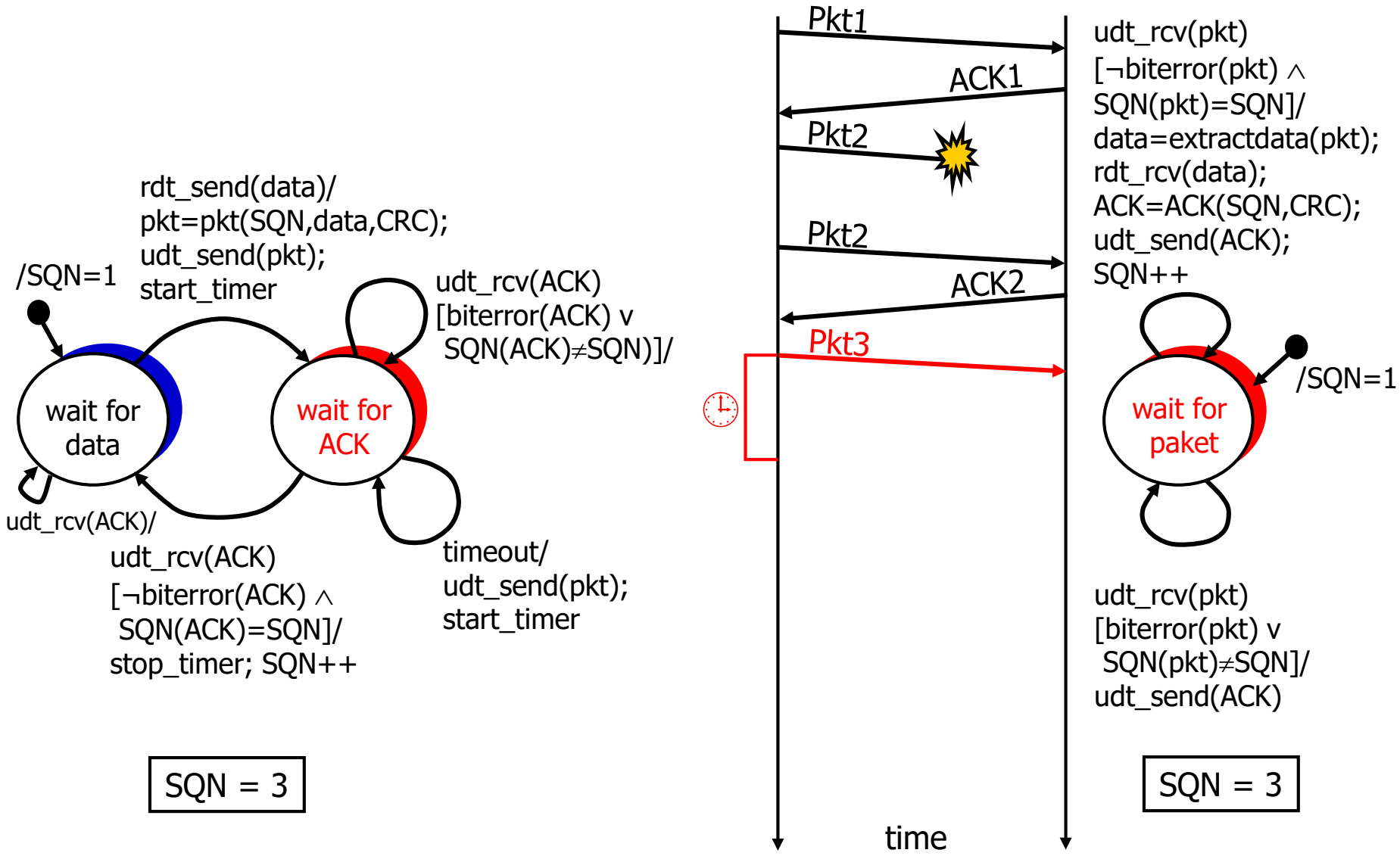


Stop-and-Wait: Verlust eines ACKs

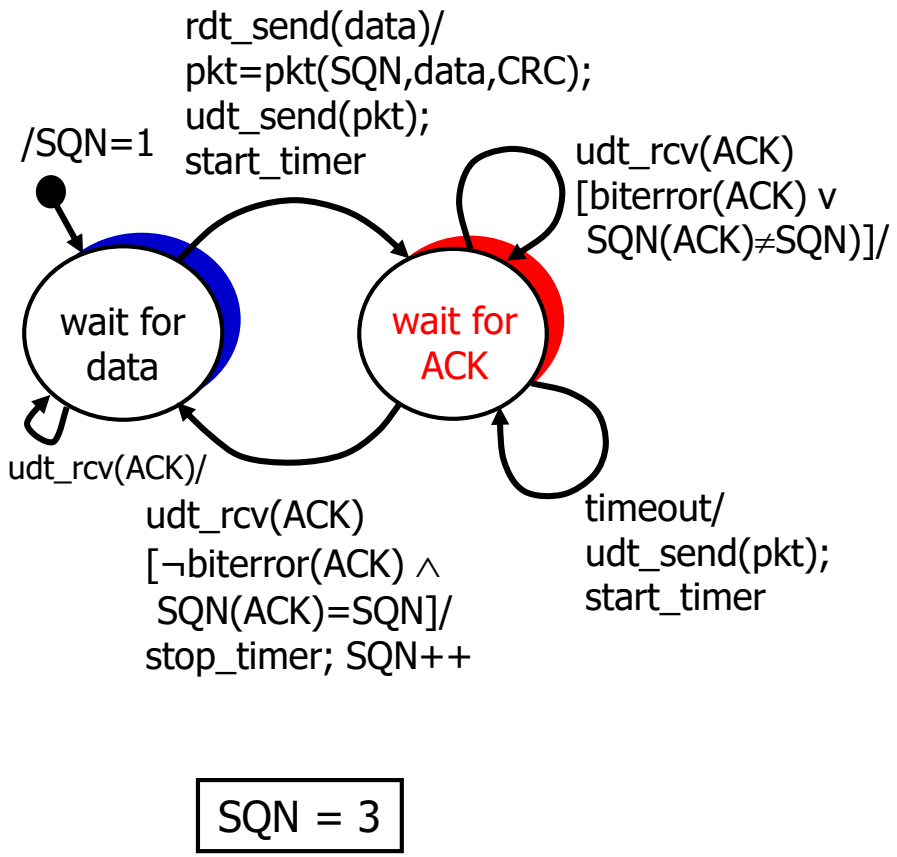


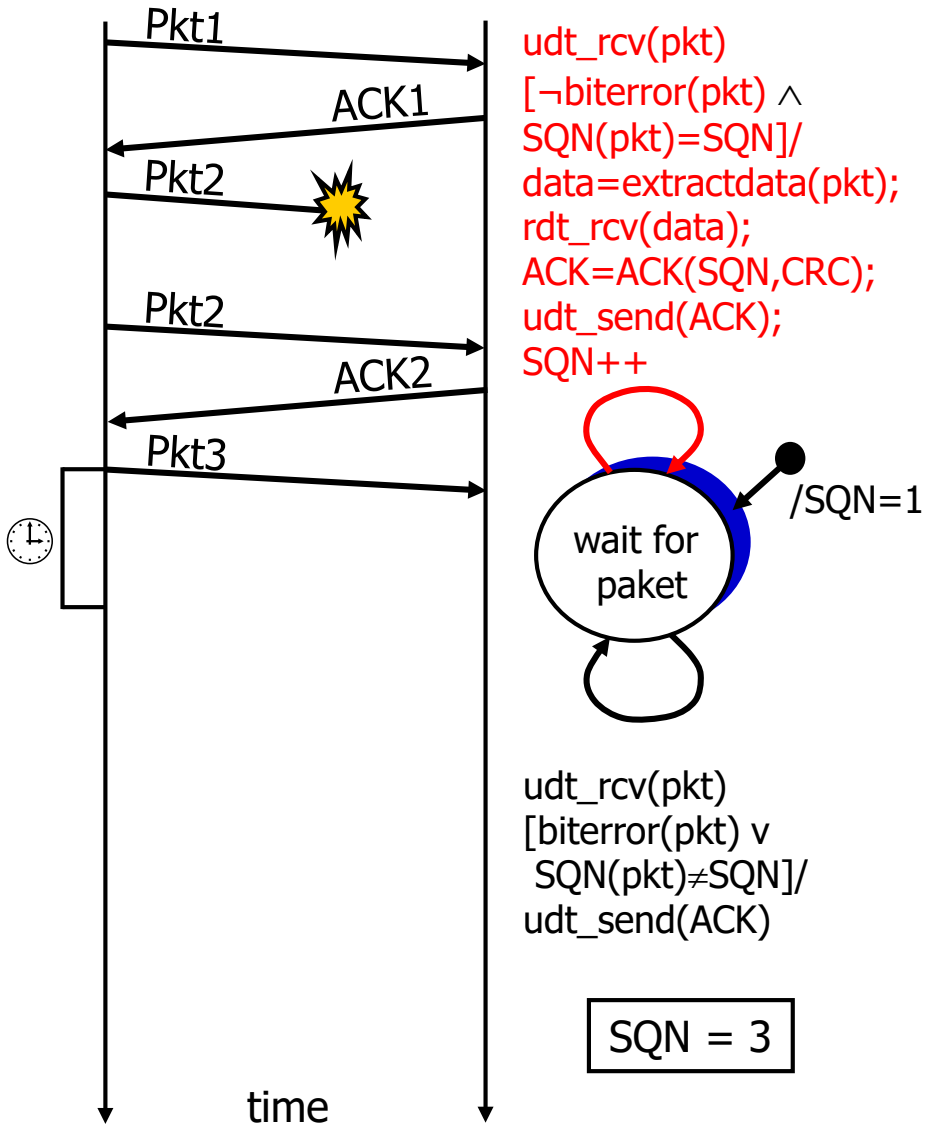


Stop-and-Wait: Verlust eines ACKs

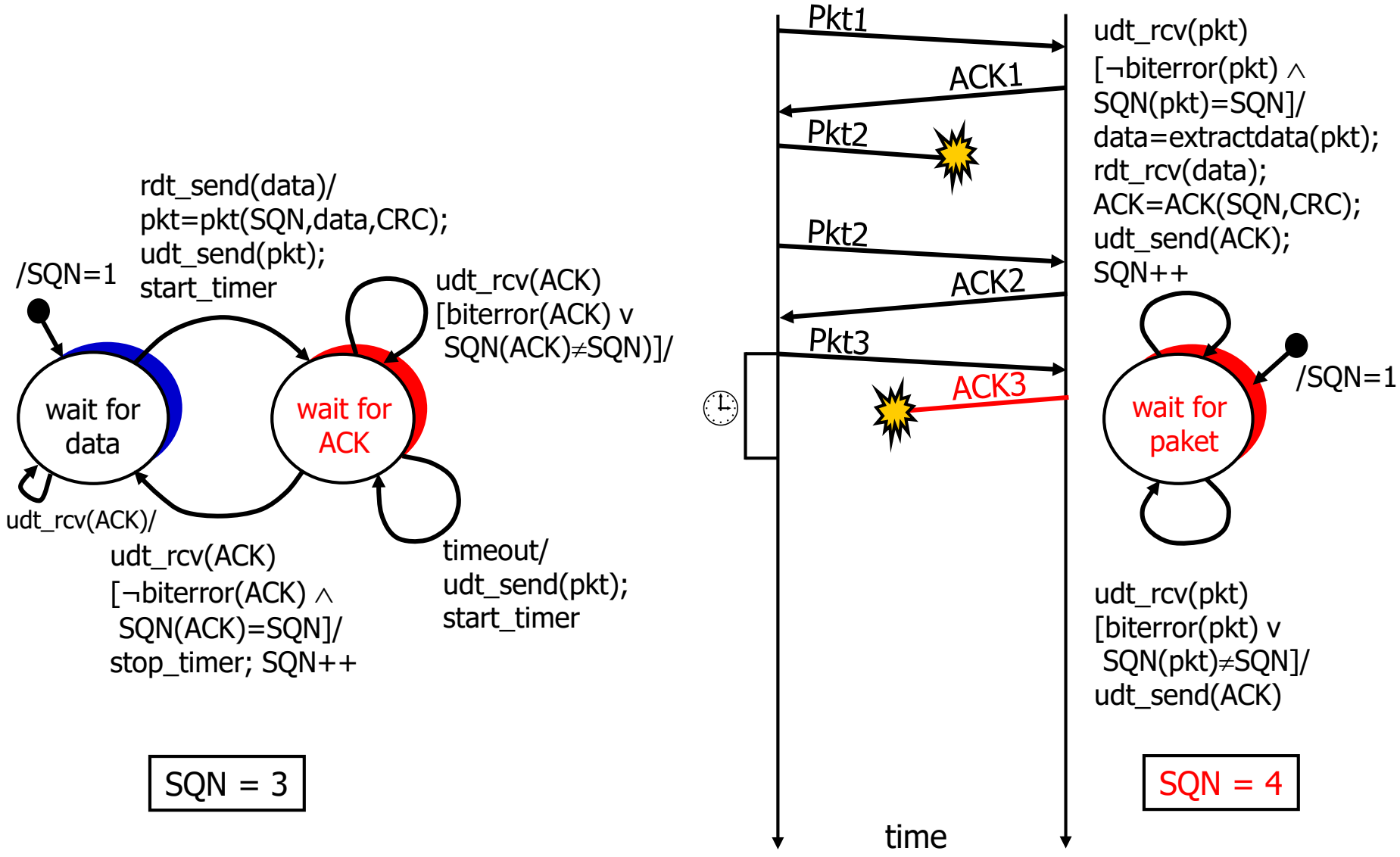


Stop-and-Wait: Verlust eines ACKs

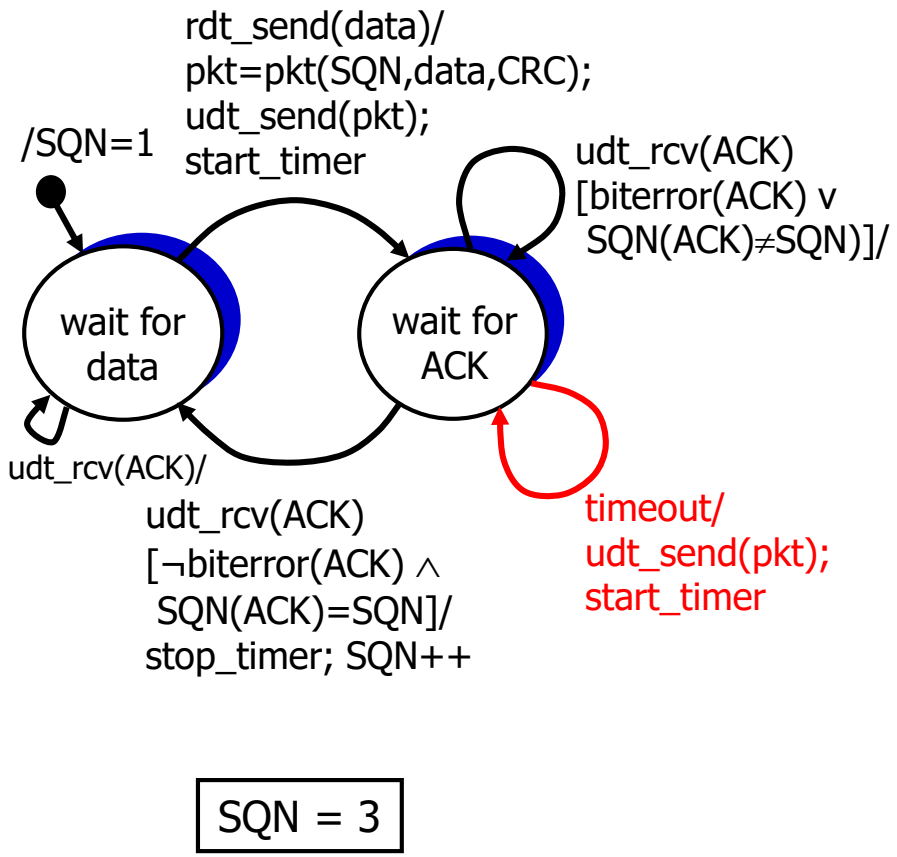


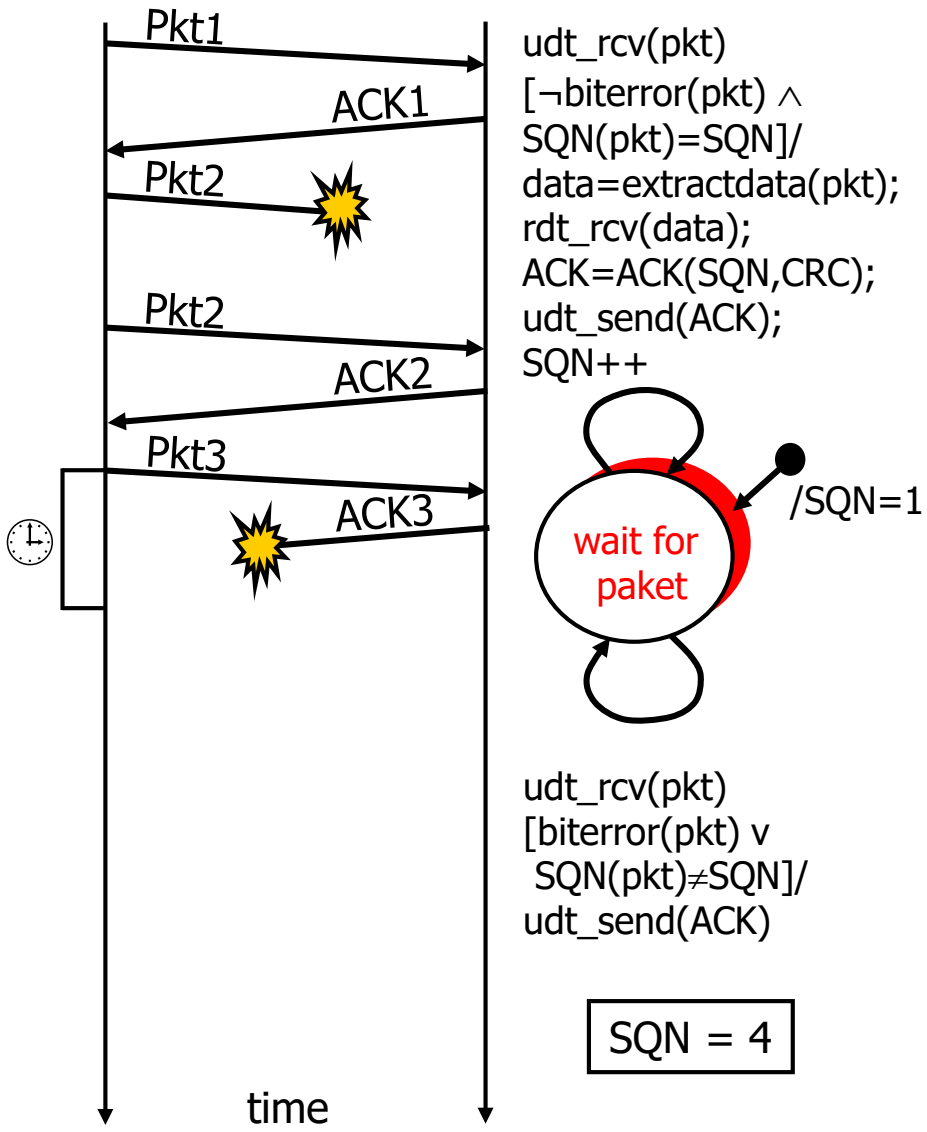


Stop-and-Wait: Verlust eines ACKs

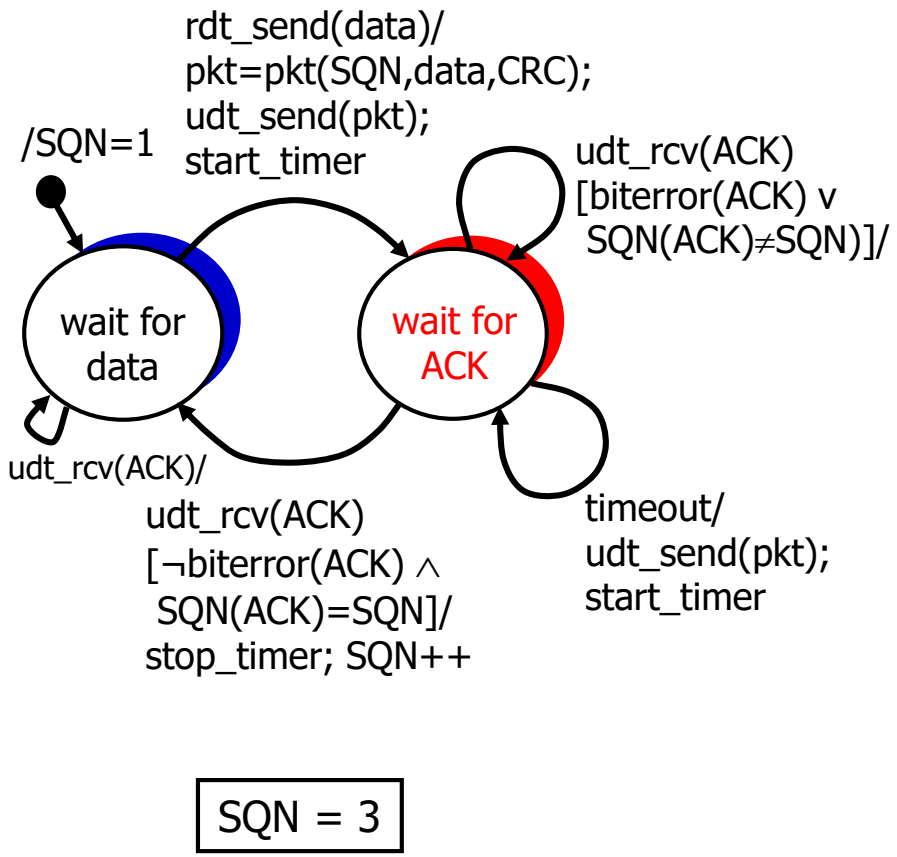


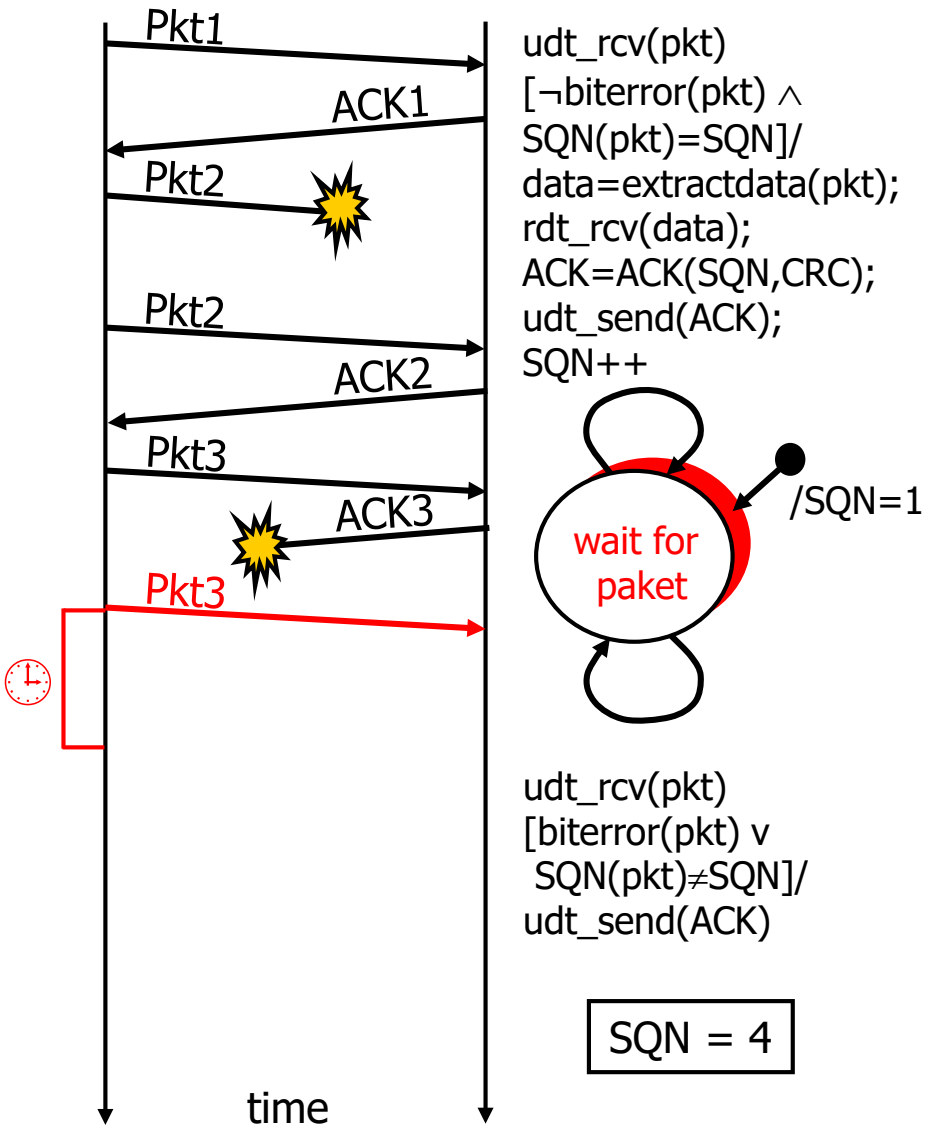
Stop-and-Wait: Verlust eines ACKs



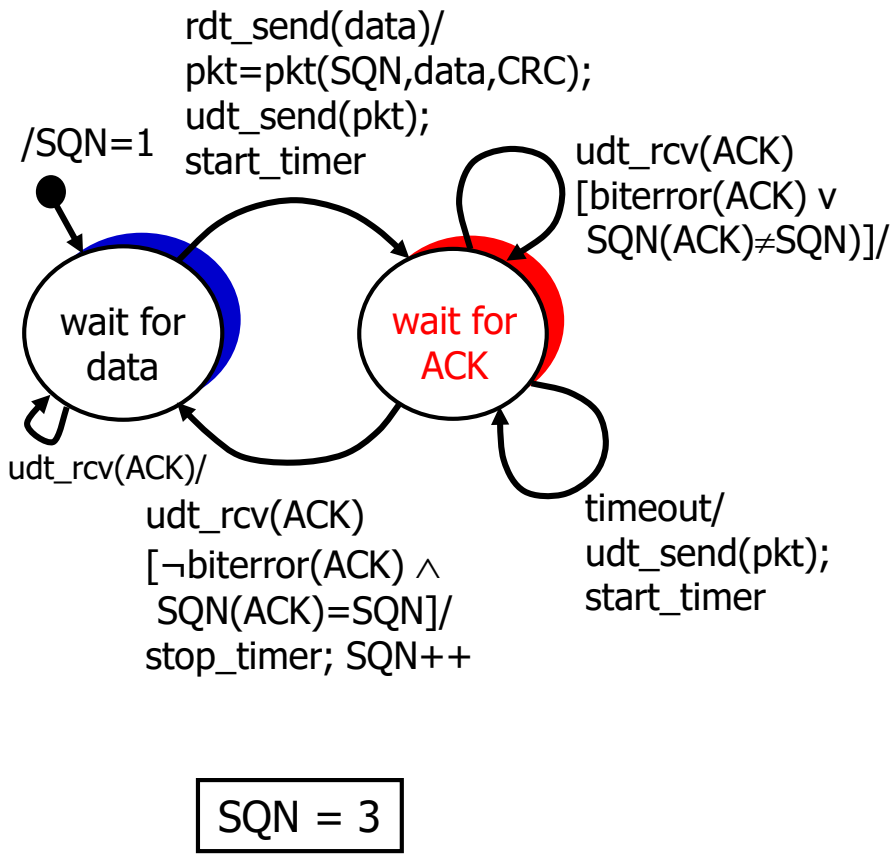


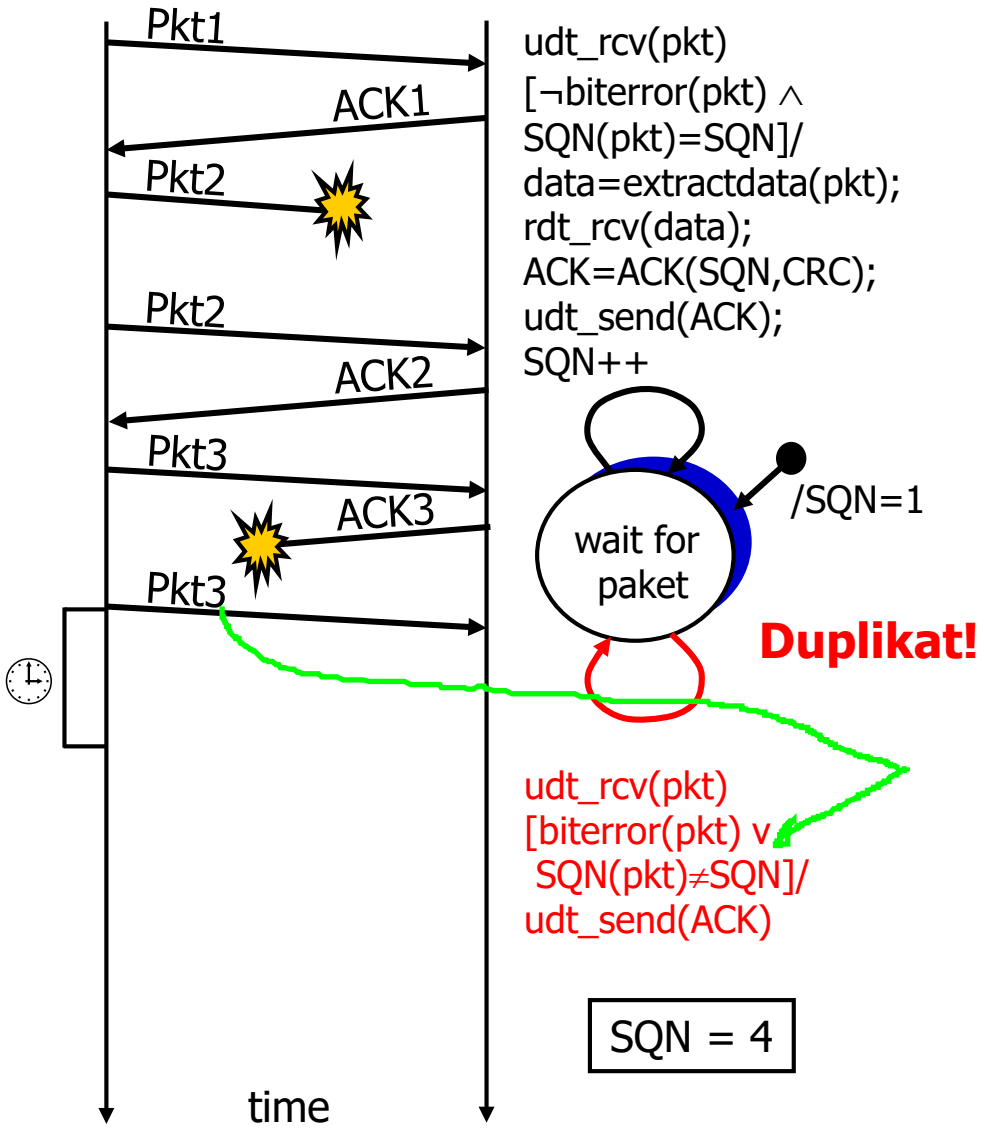
Stop-and-Wait: Verlust eines ACKs



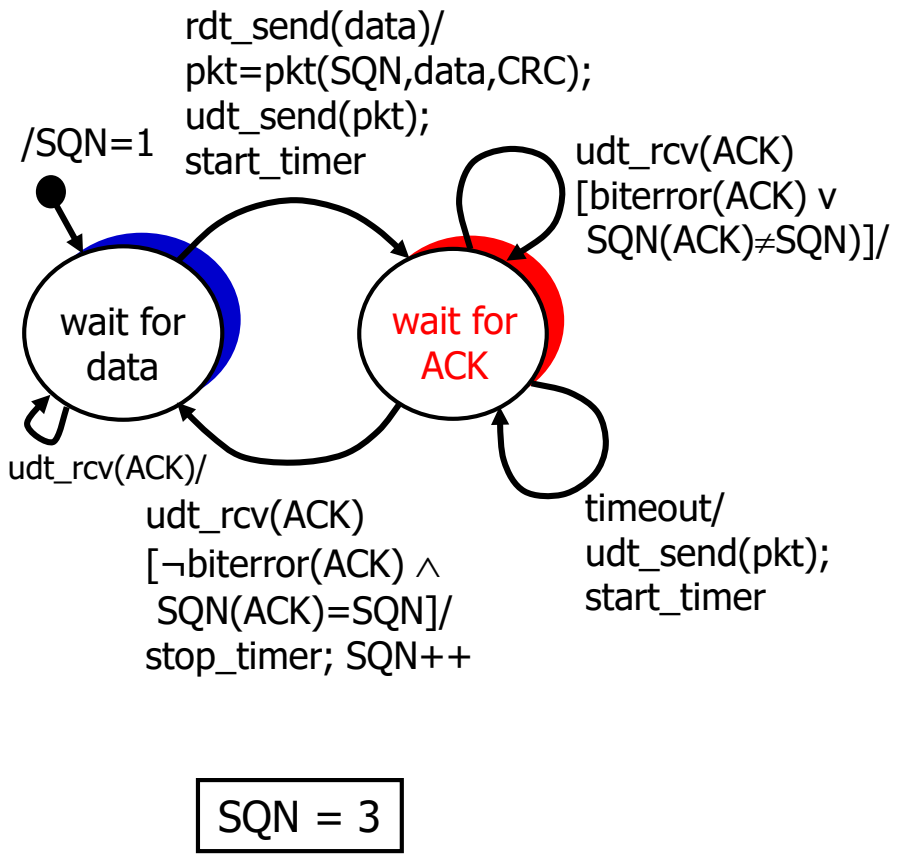


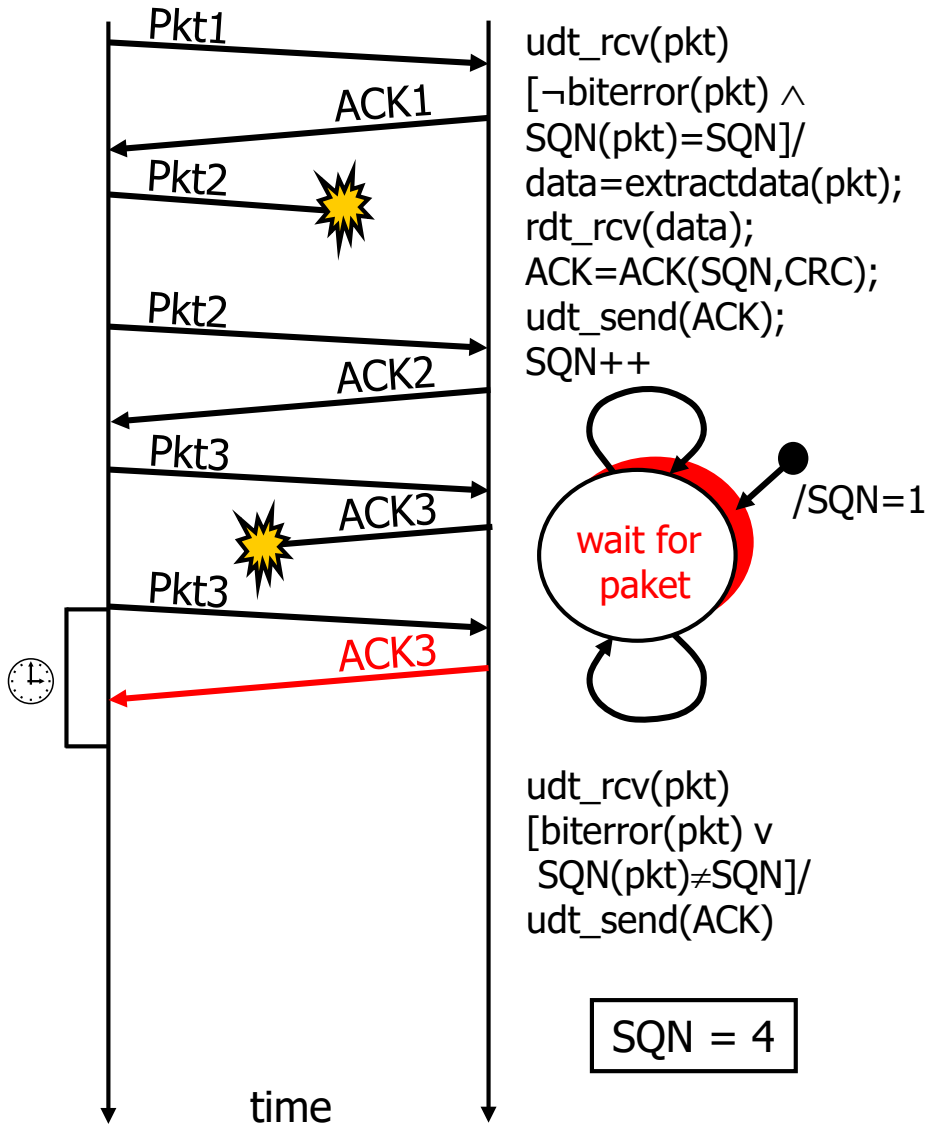
Stop-and-Wait: Verlust eines ACKs



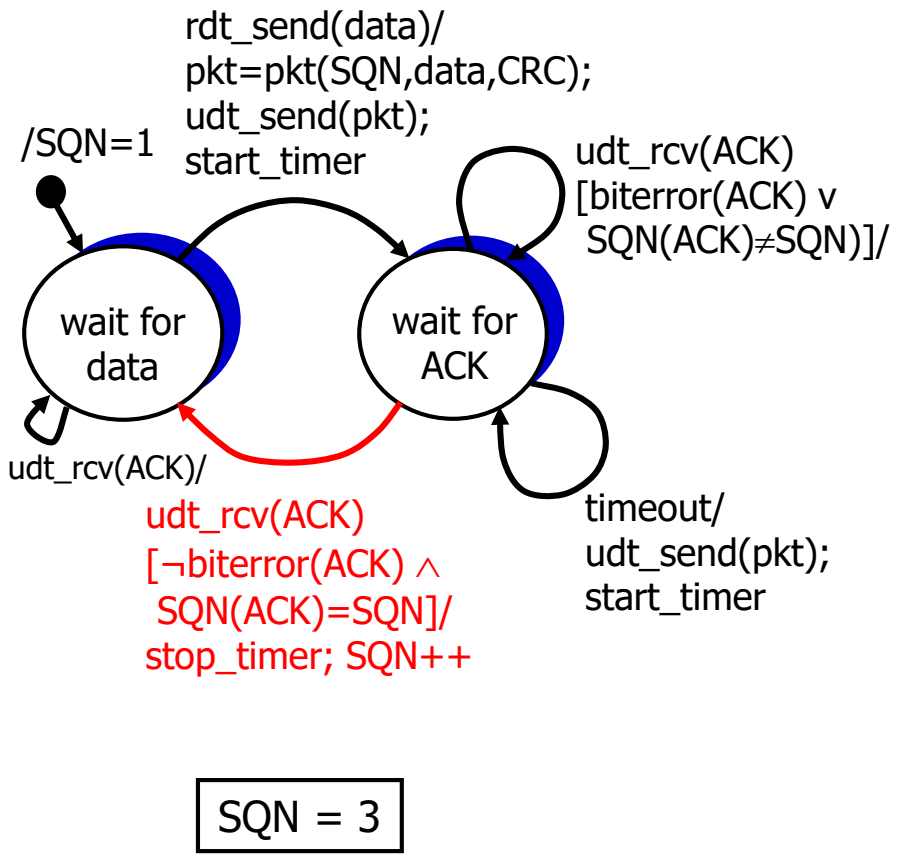


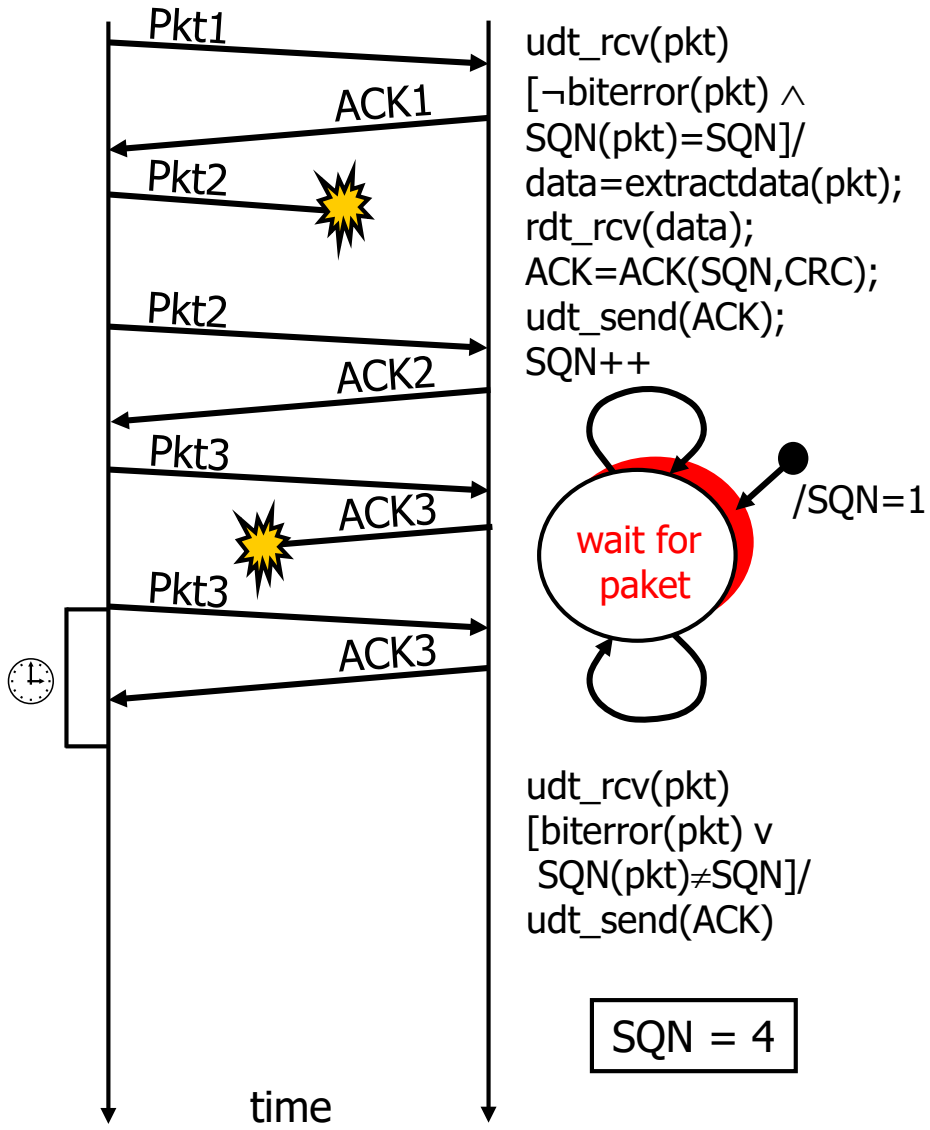
Stop-and-Wait: Verlust eines ACKs



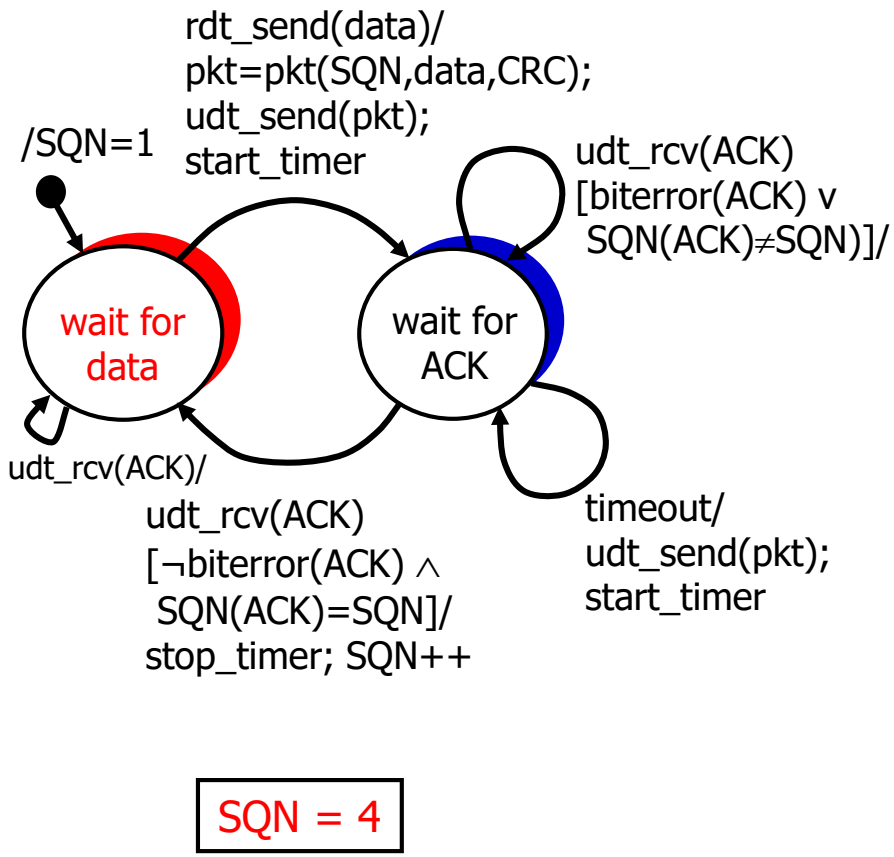


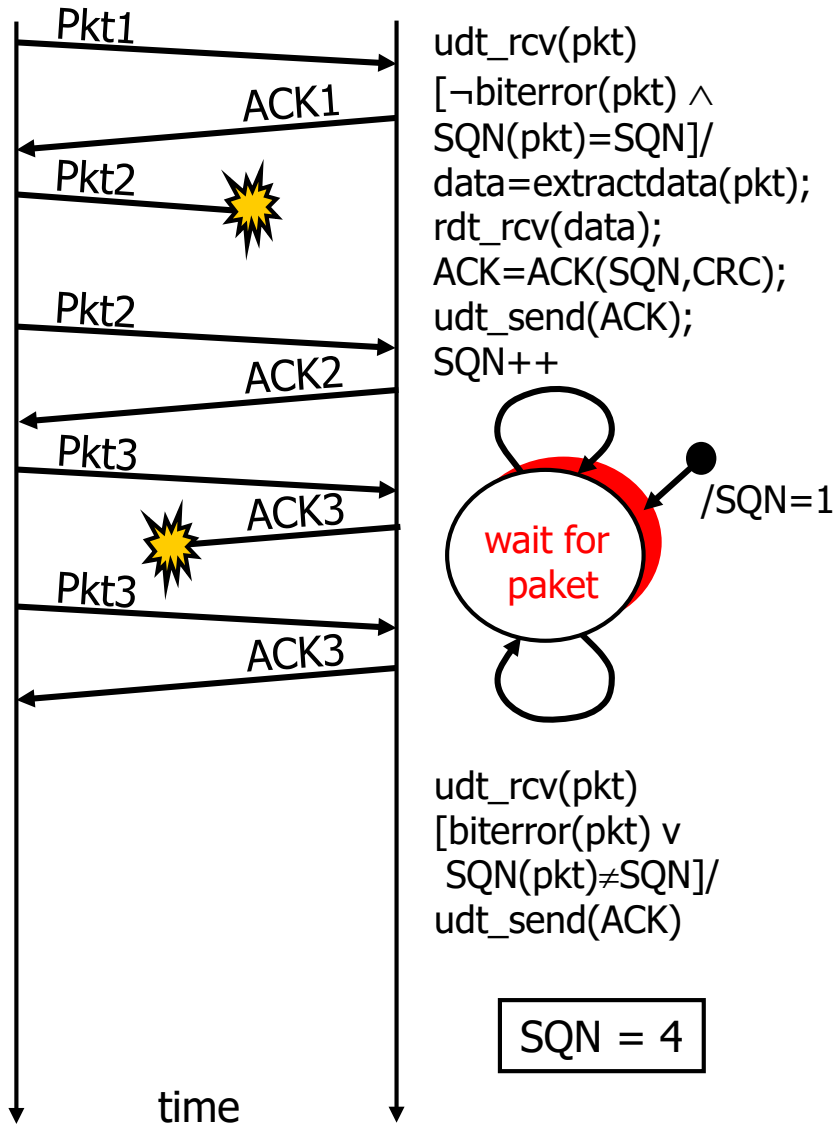
Stop-and-Wait: Verlust eines ACKs



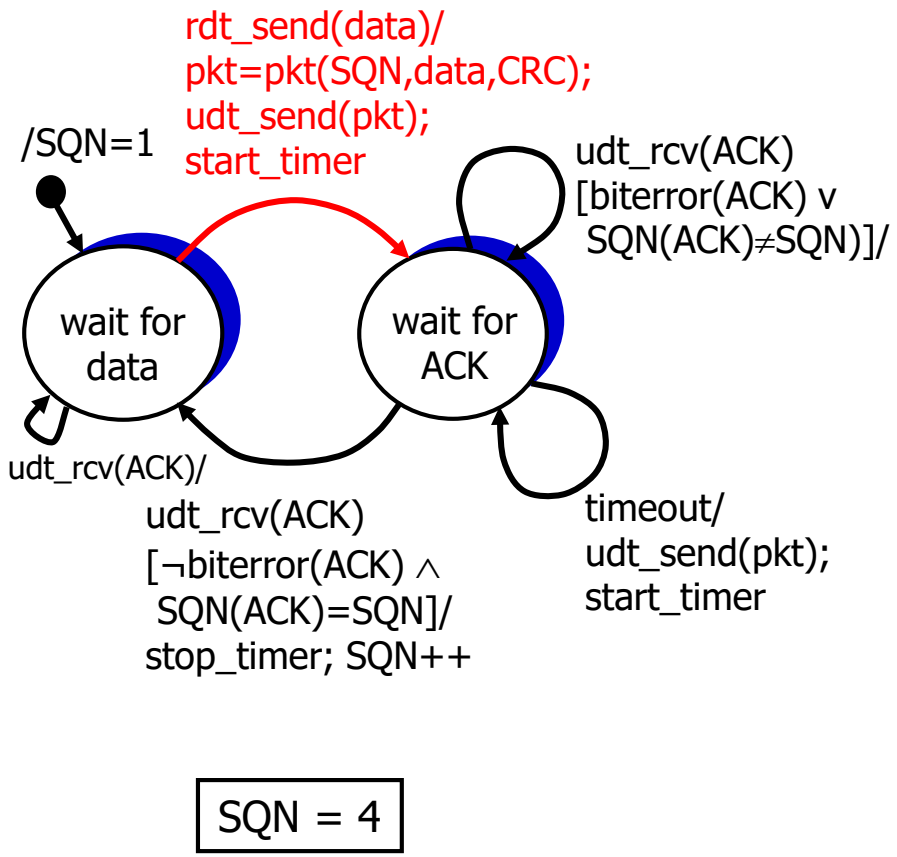


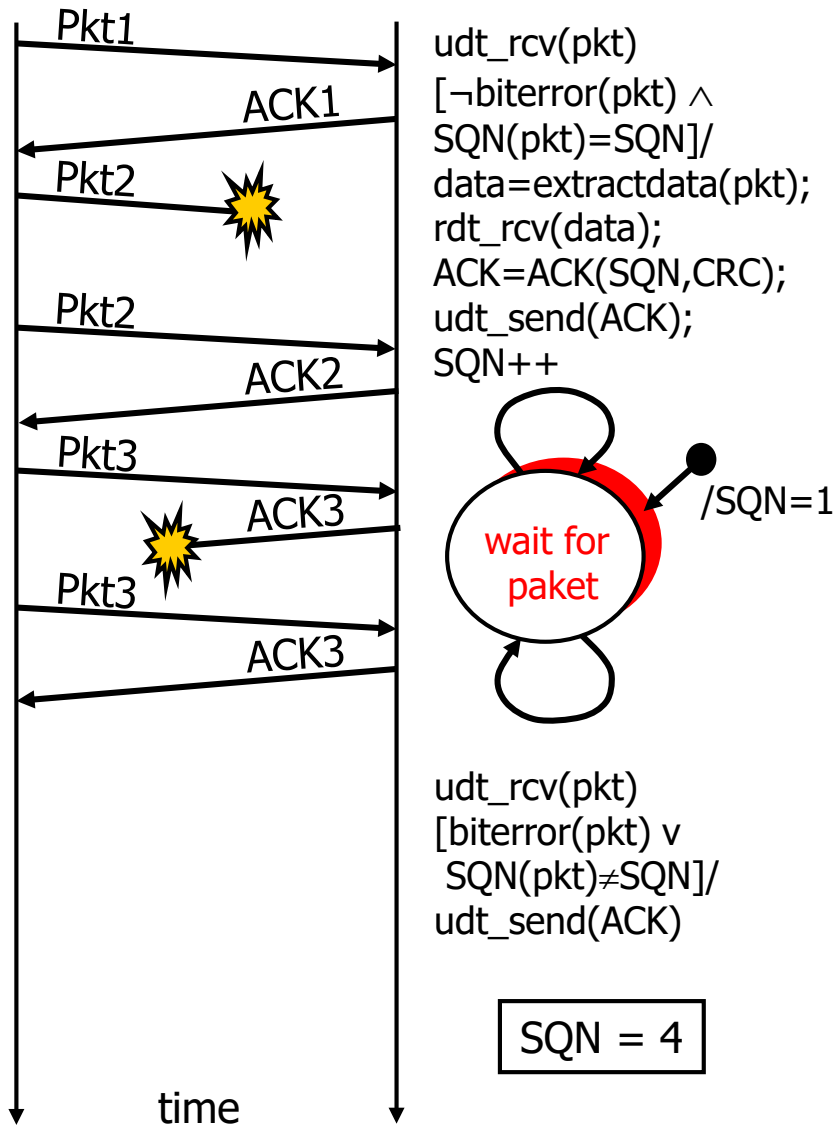
Stop-and-Wait: Verlust eines ACKs



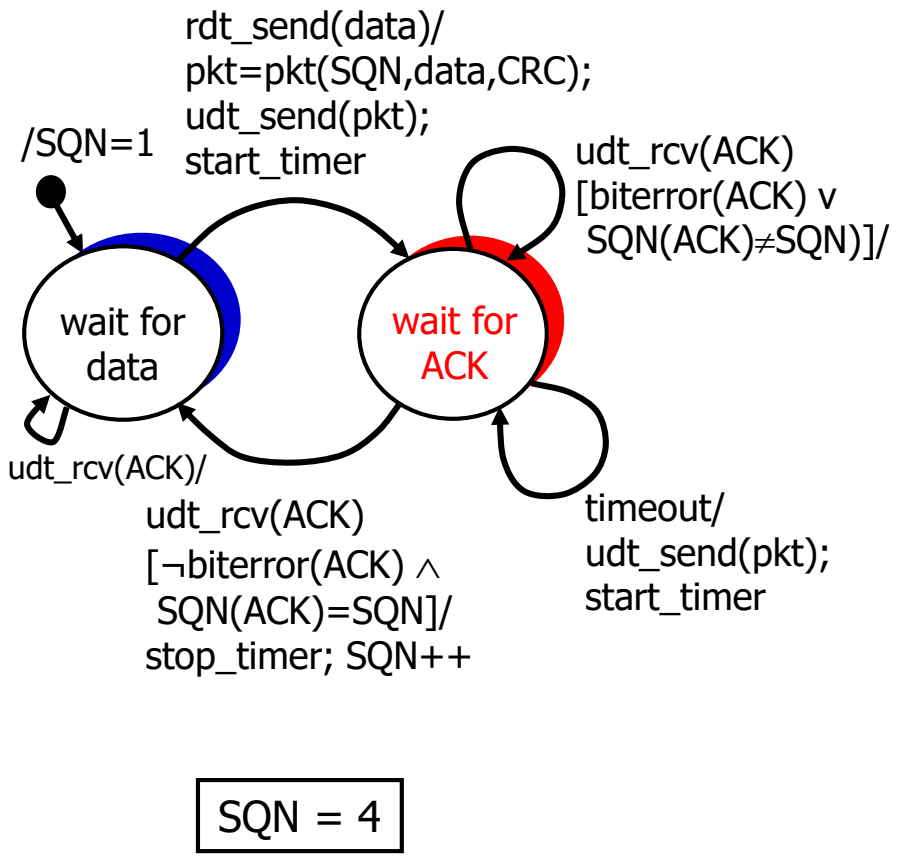


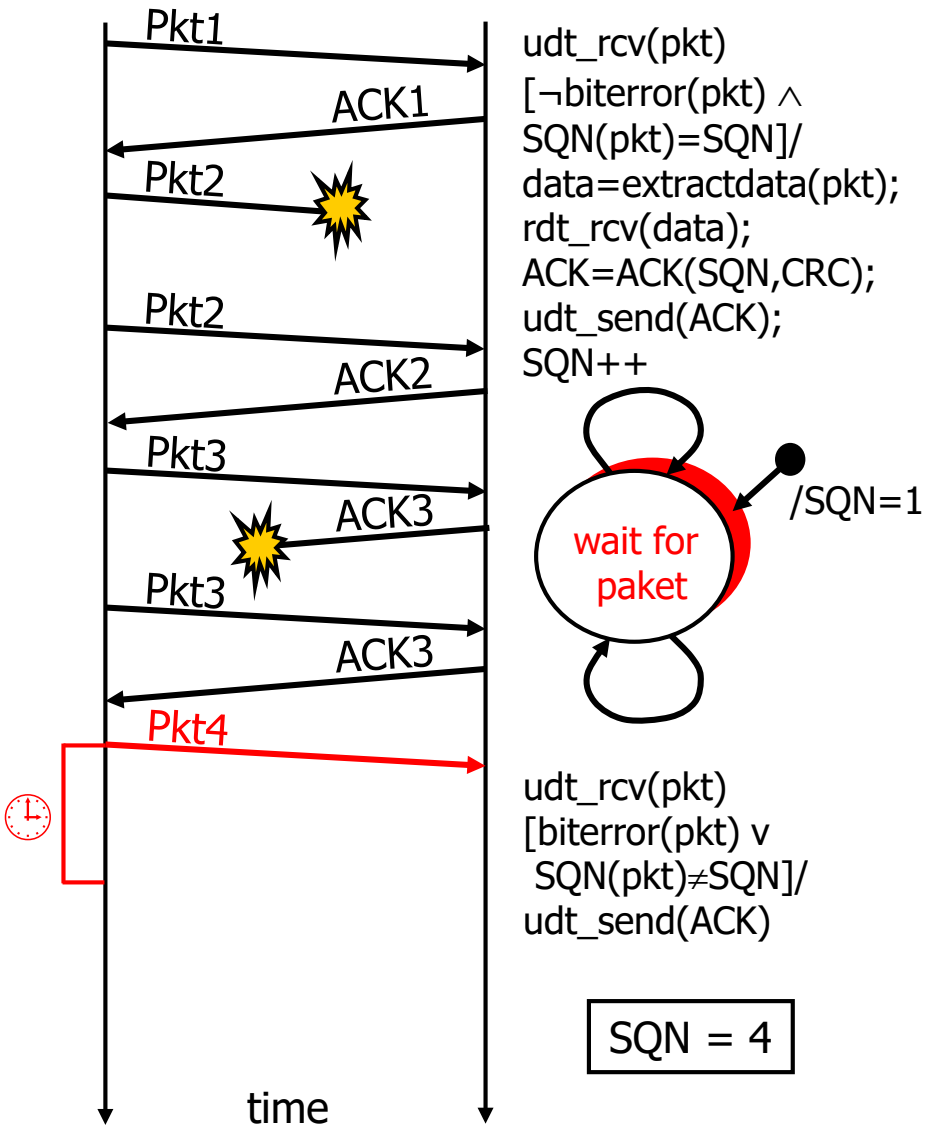
Stop-and-Wait: verzögertes ACK



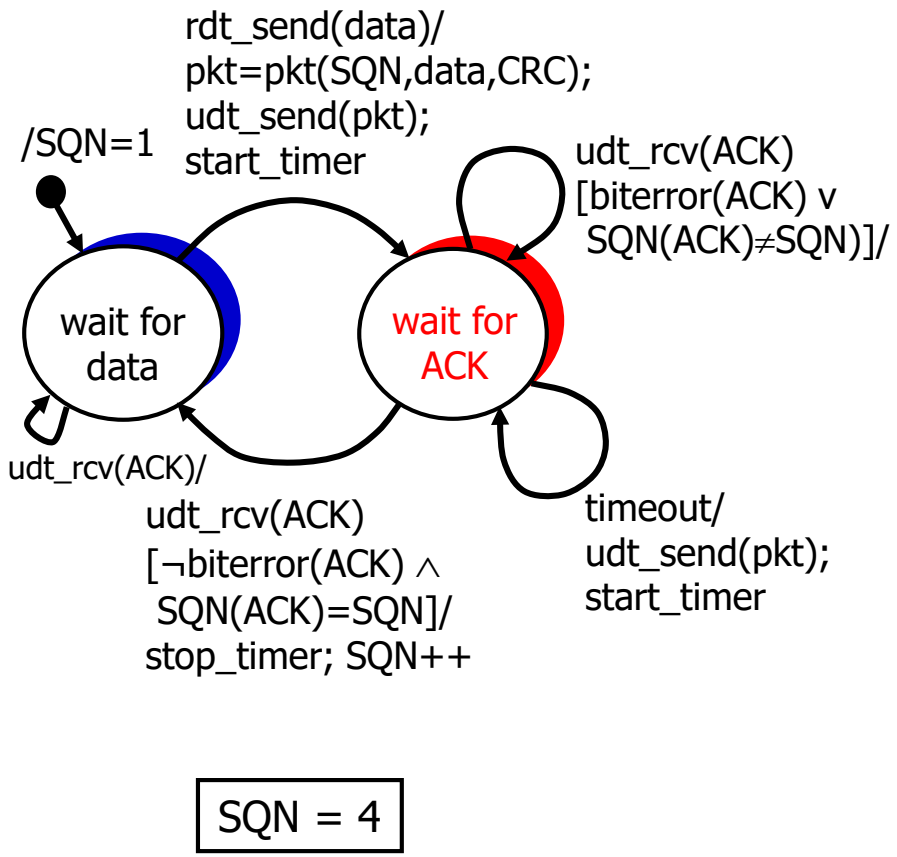


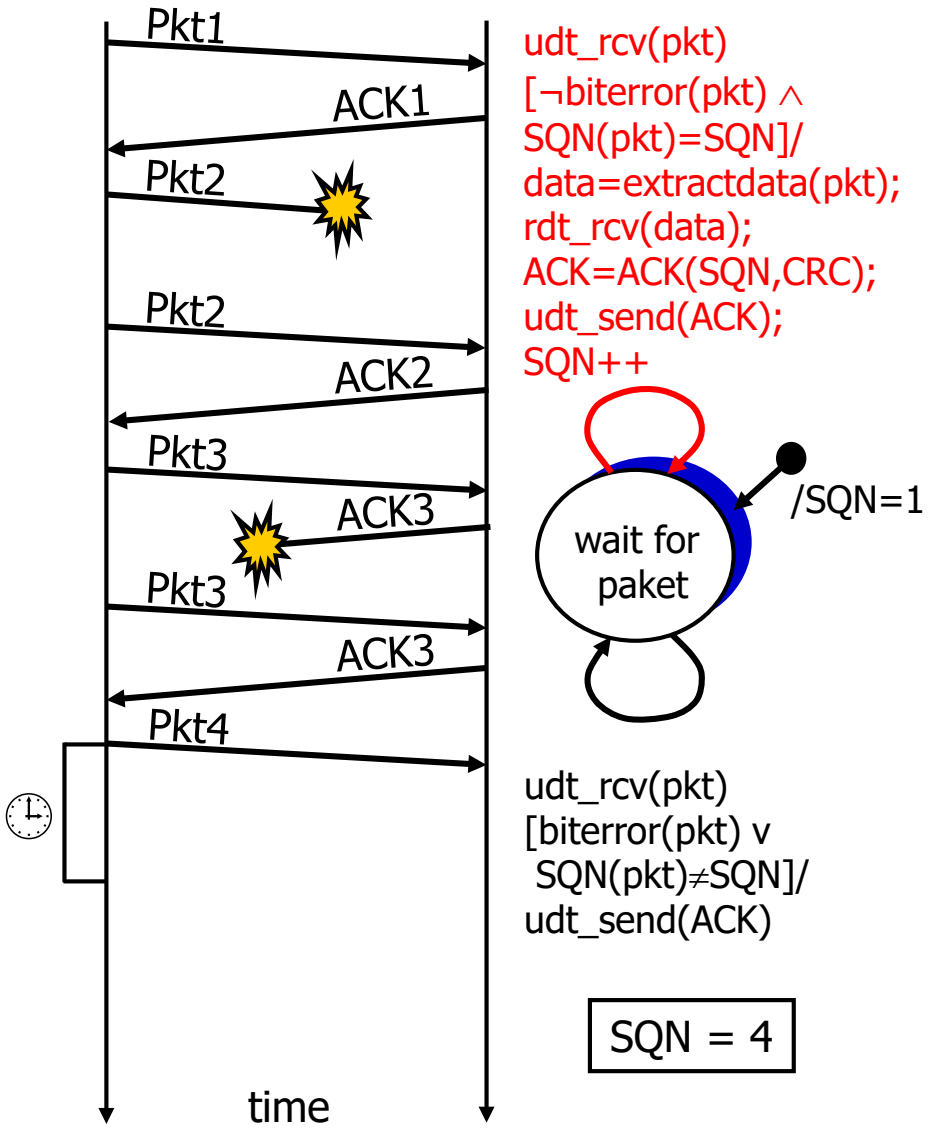
Stop-and-Wait: verzögertes ACK



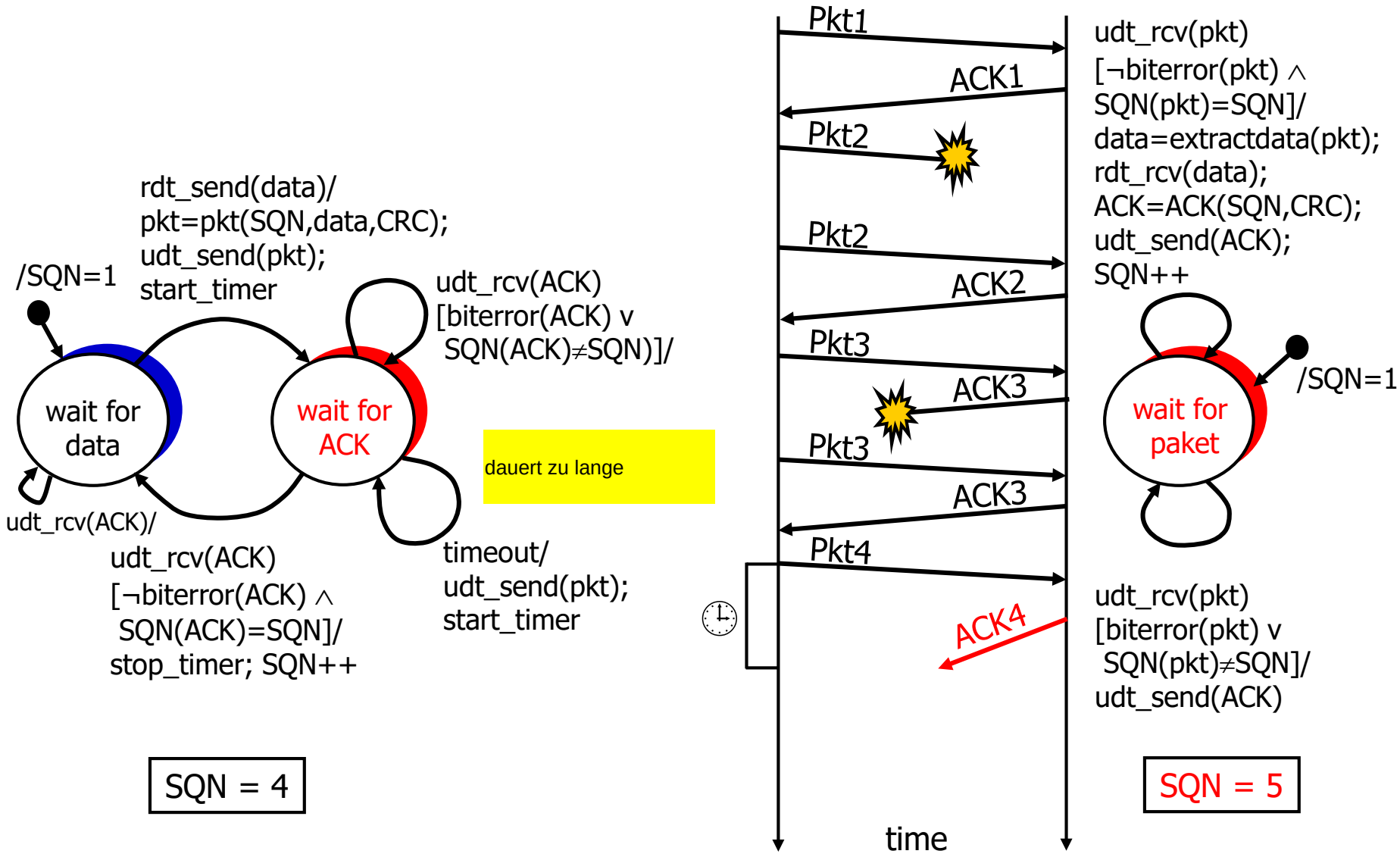


Stop-and-Wait: verzögertes ACK

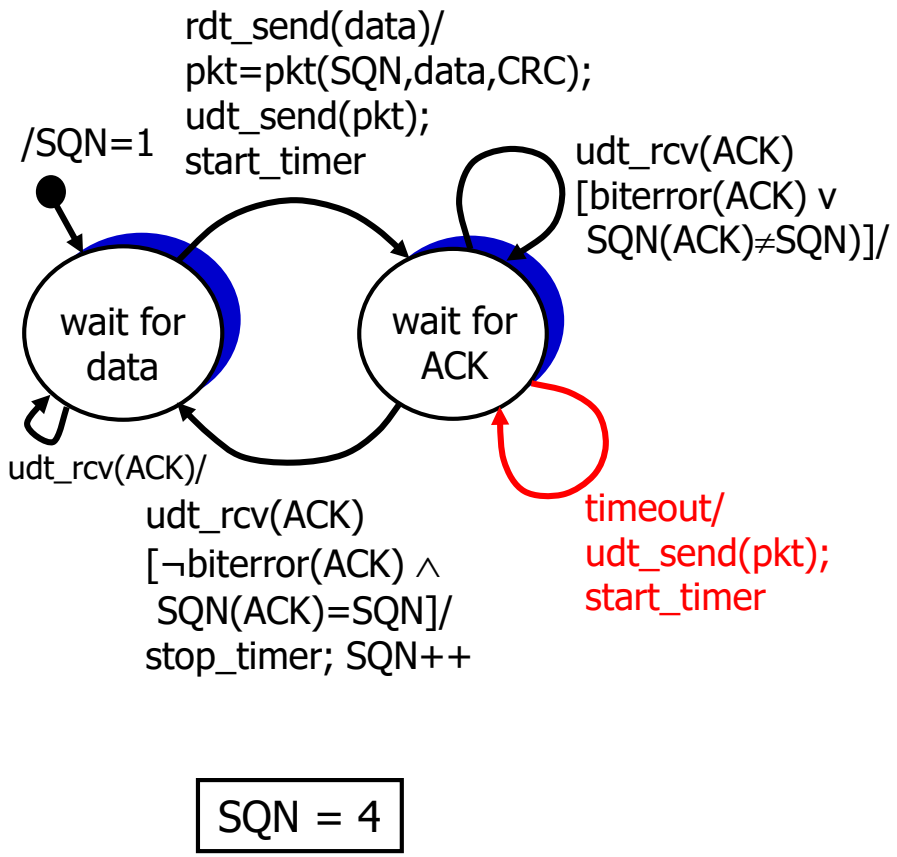


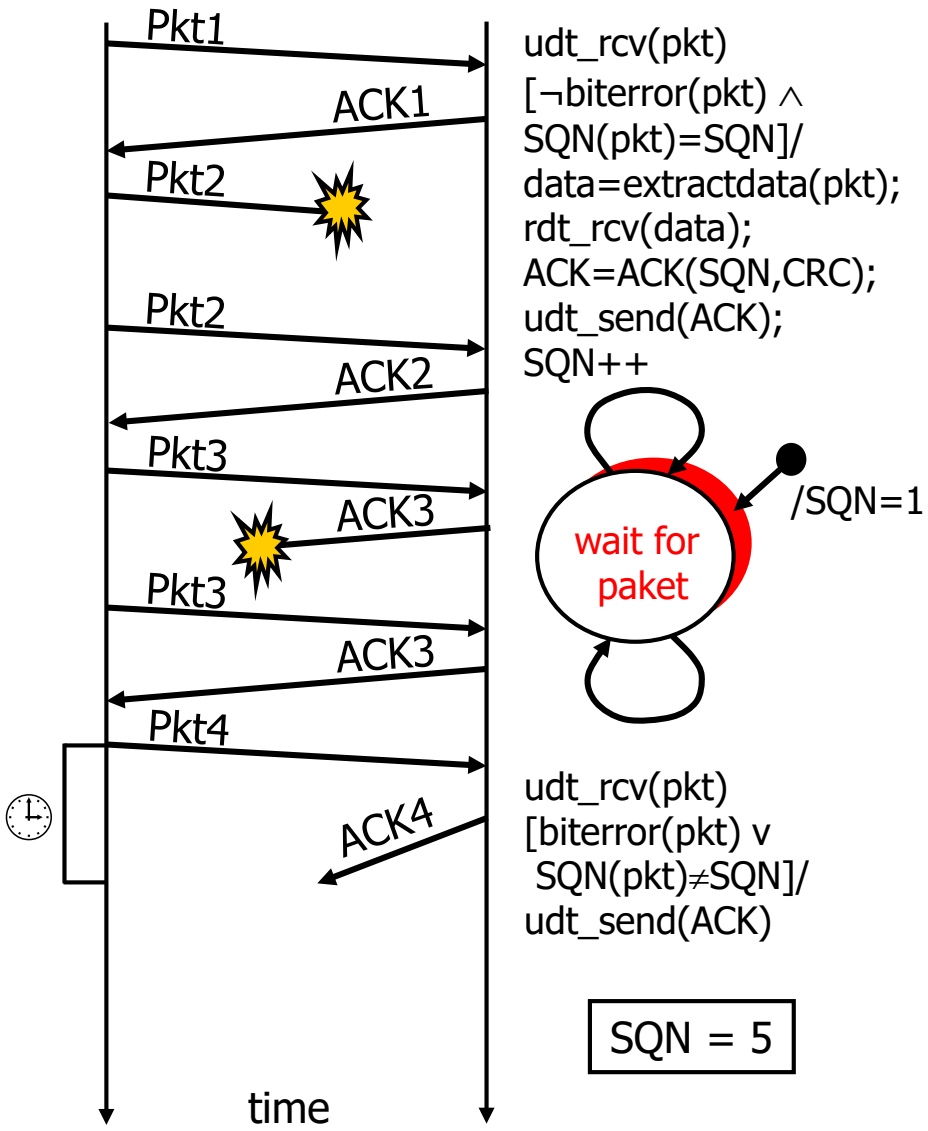


Stop-and-Wait: verzögertes ACK

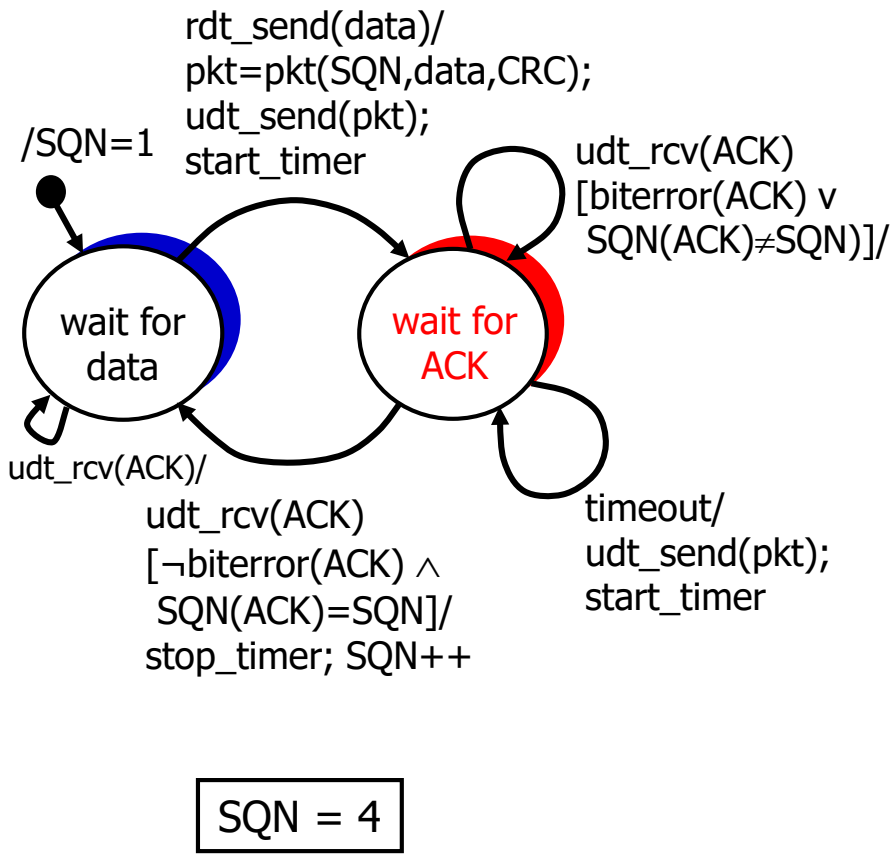


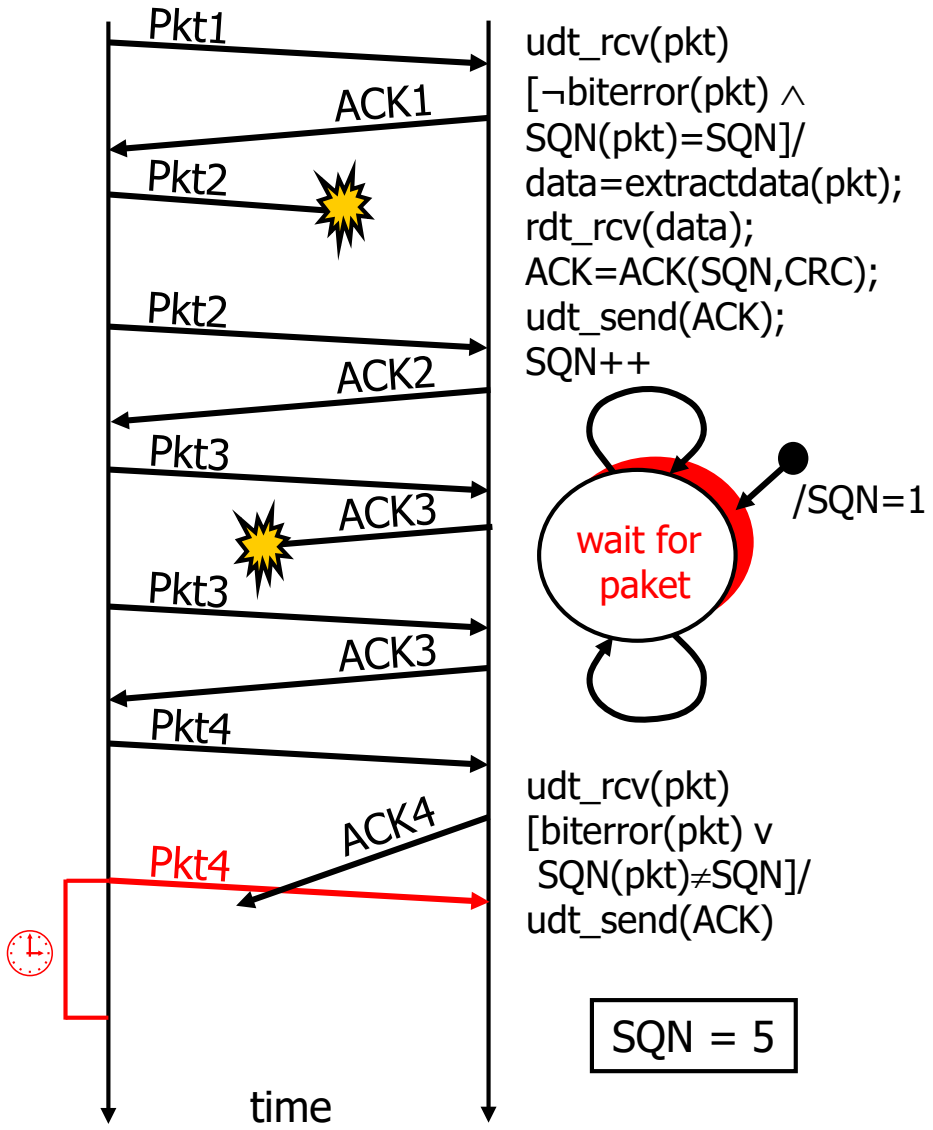
Stop-and-Wait: verzögertes ACK



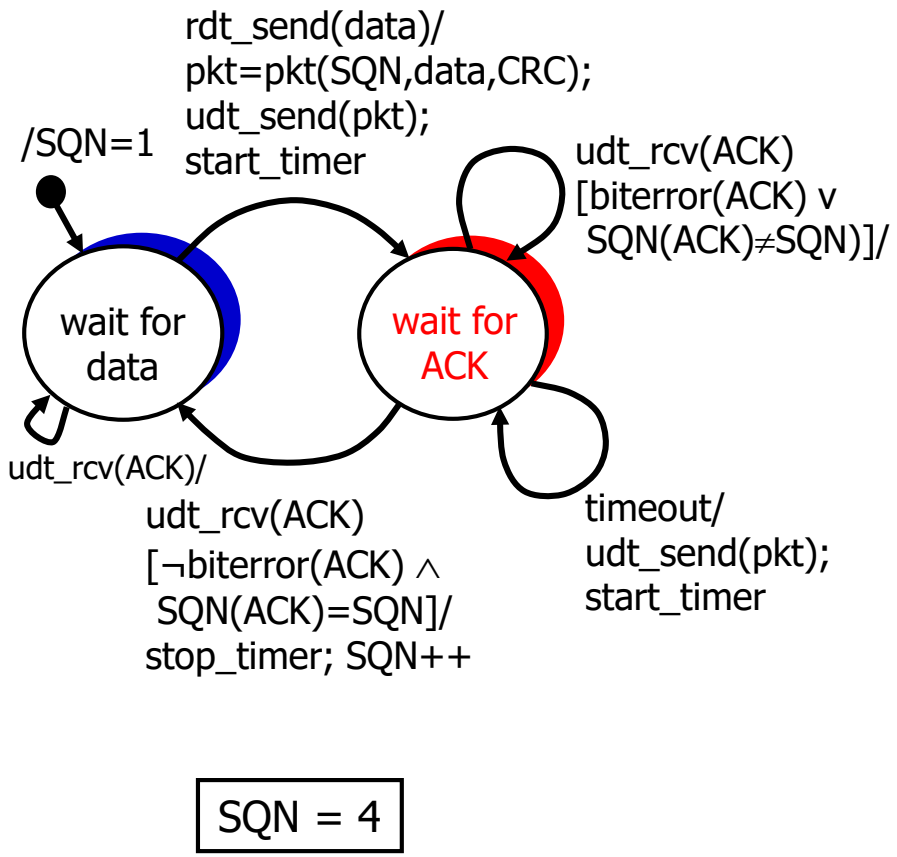


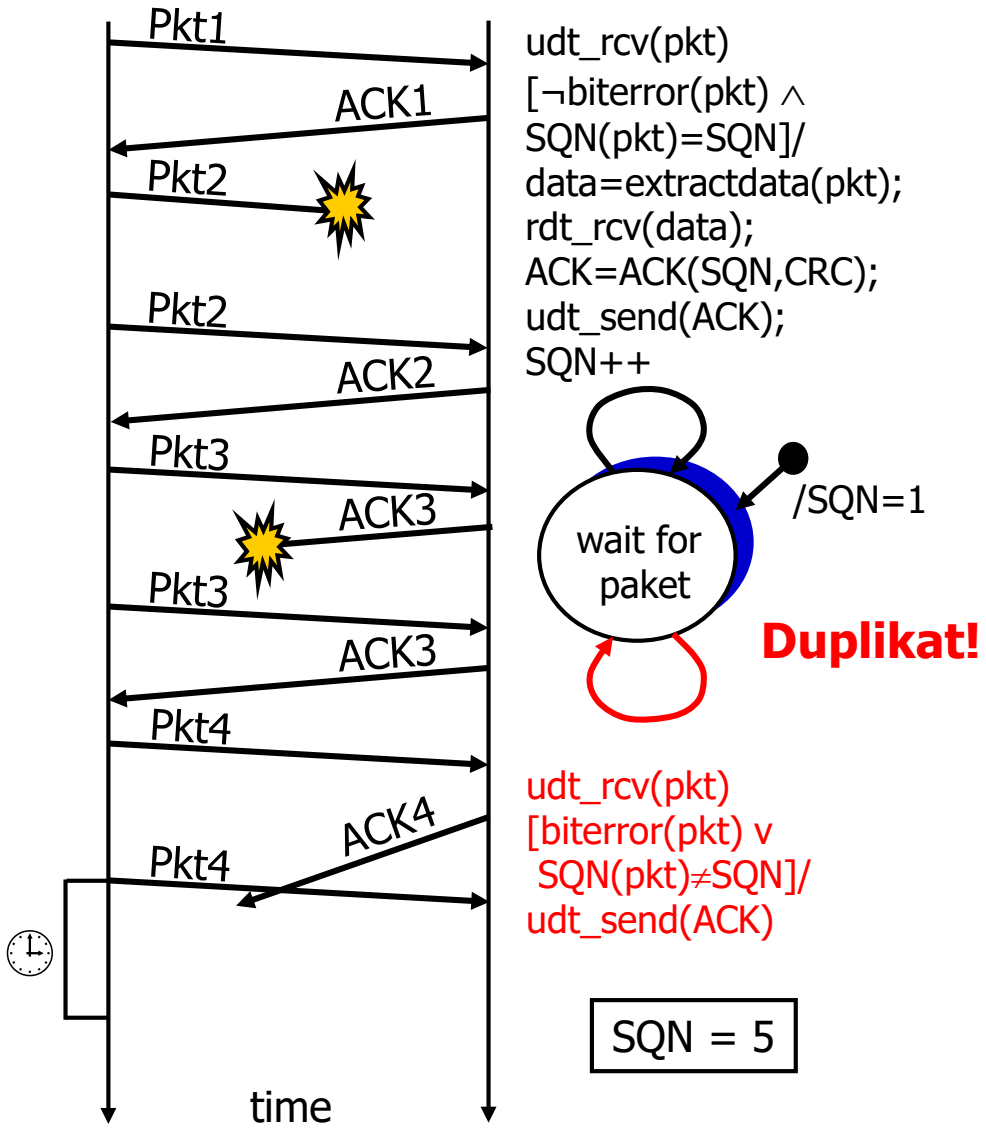
Stop-and-Wait: verzögertes ACK



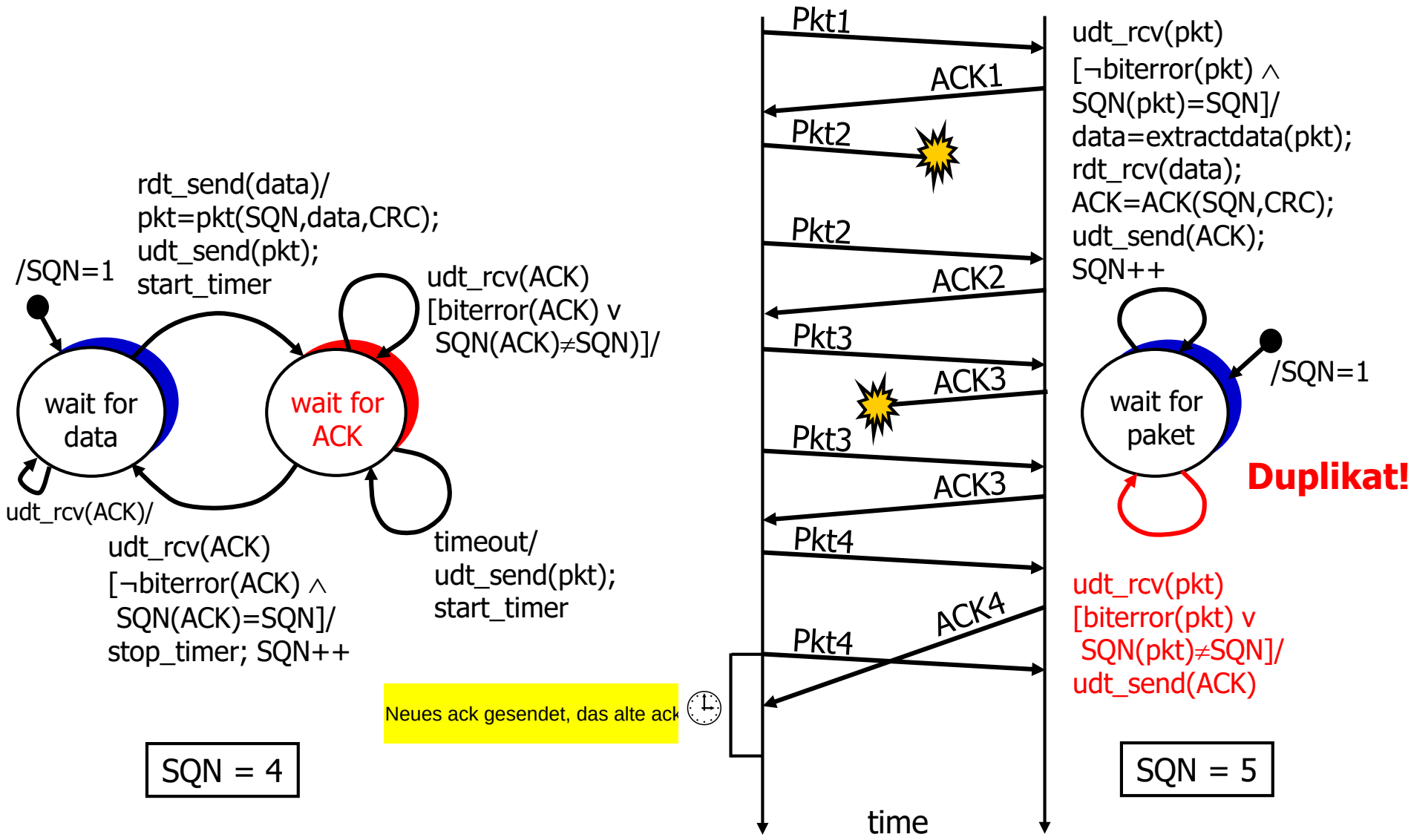


Stop-and-Wait: verzögertes ACK

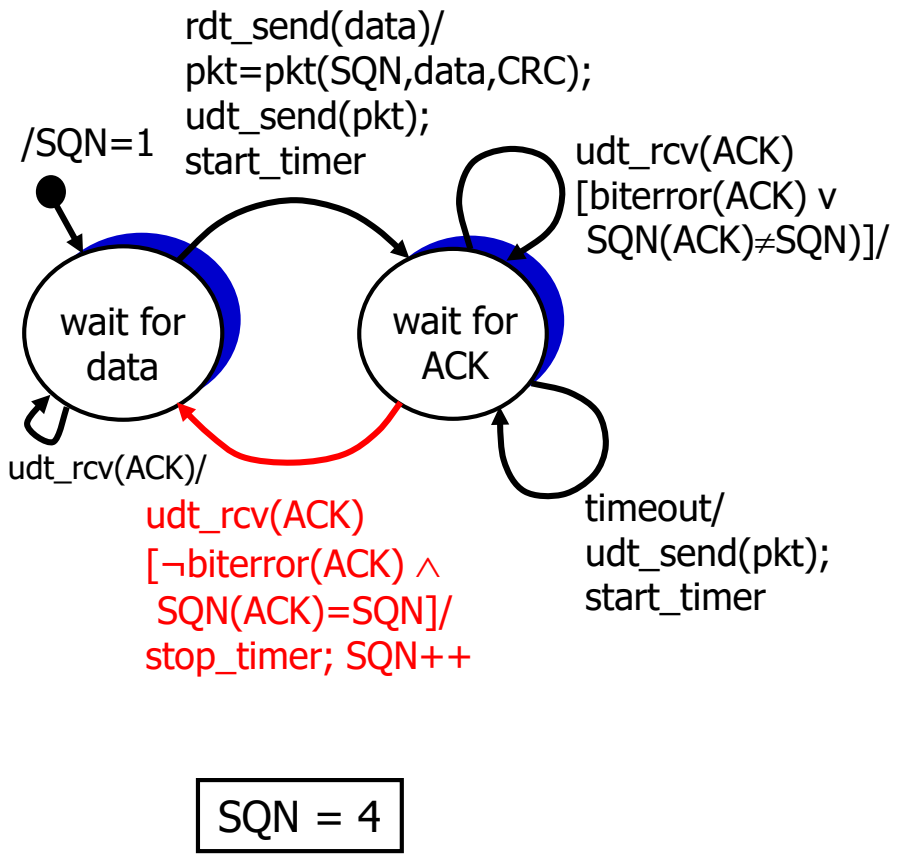


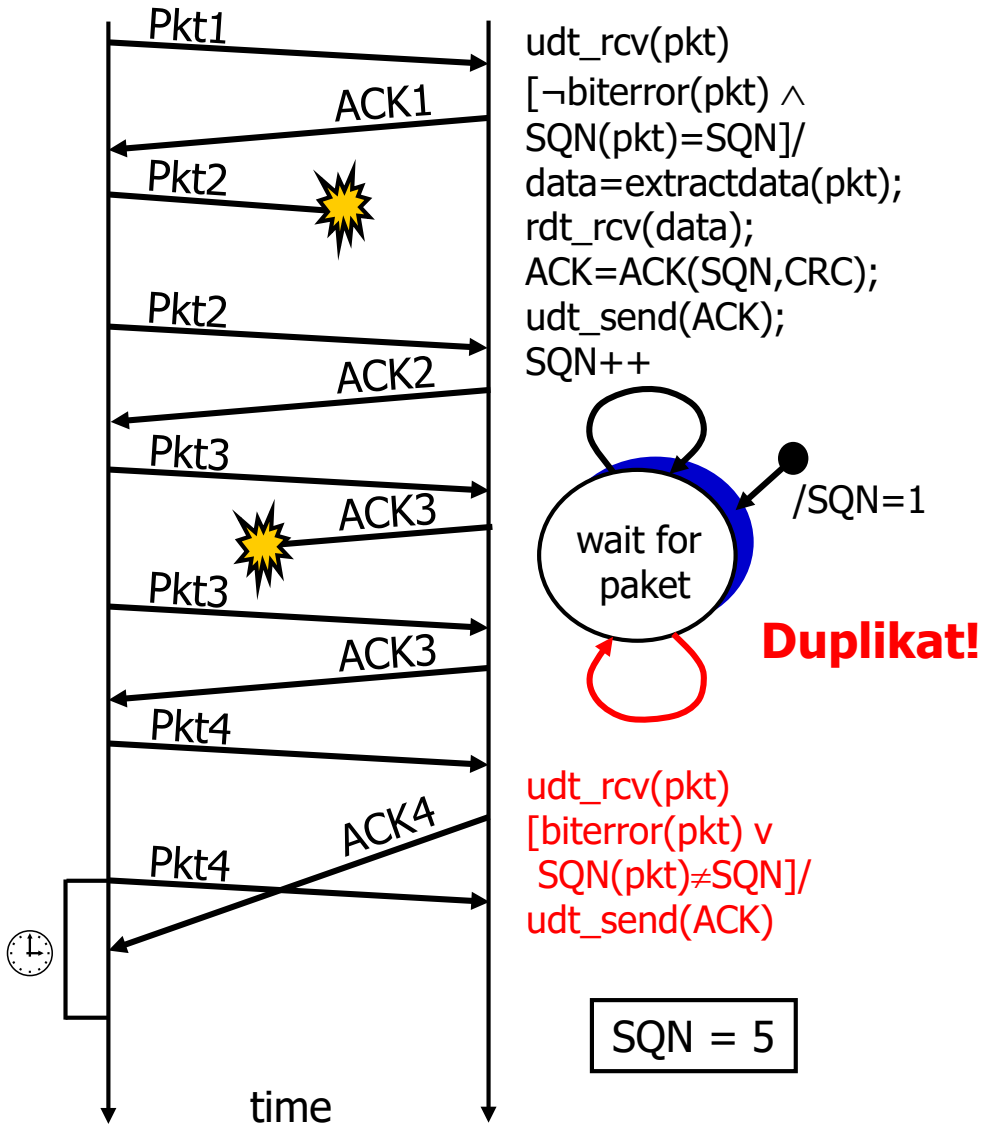


Stop-and-Wait: verzögertes ACK

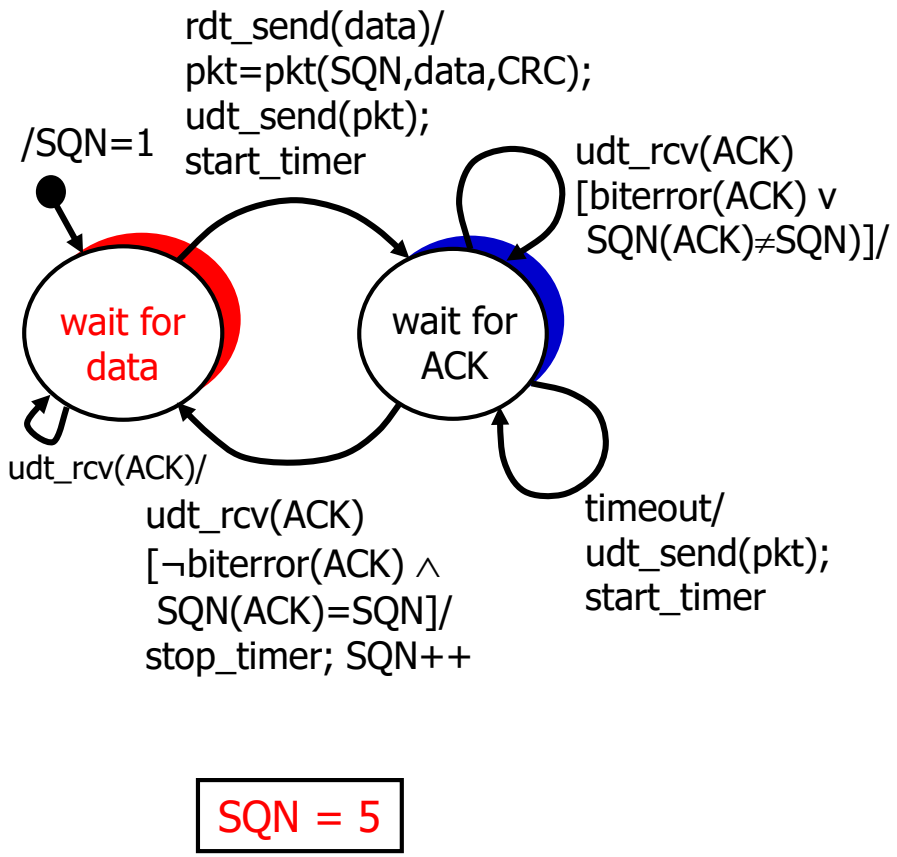


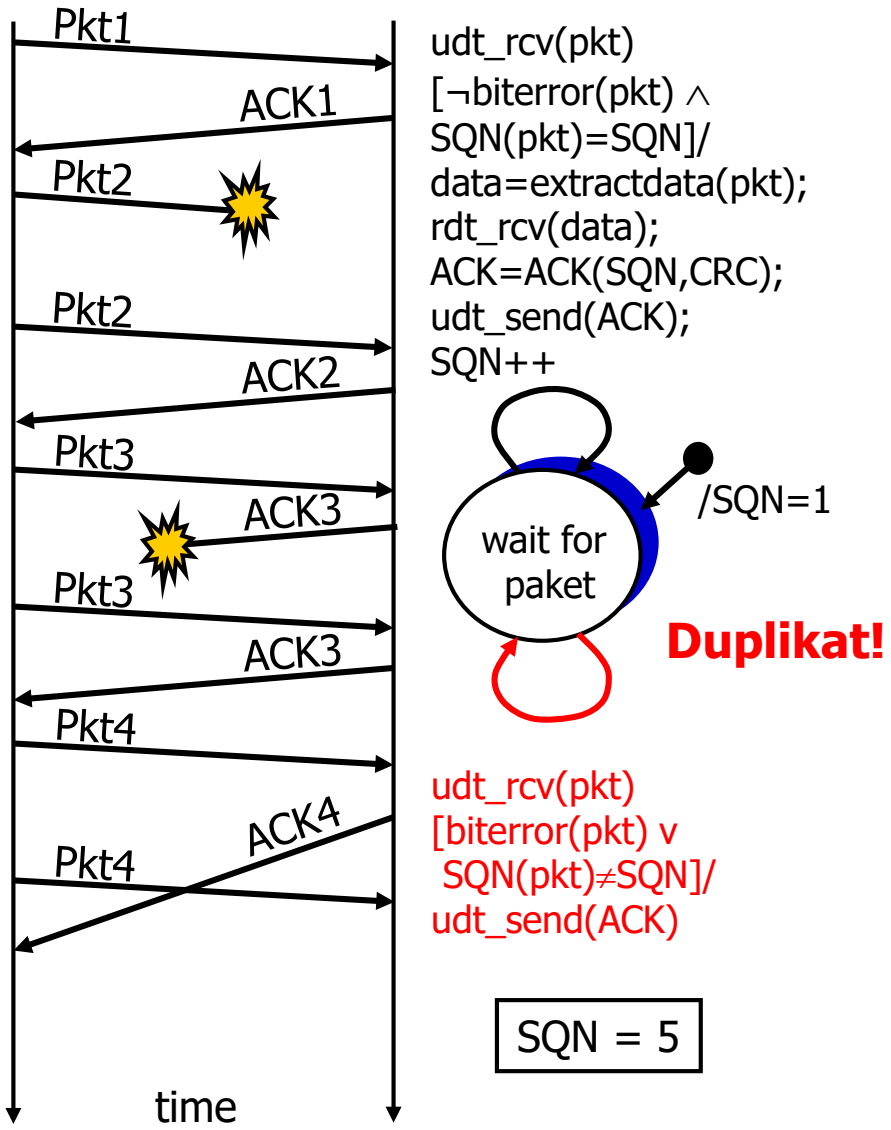
Stop-and-Wait: verzögertes ACK



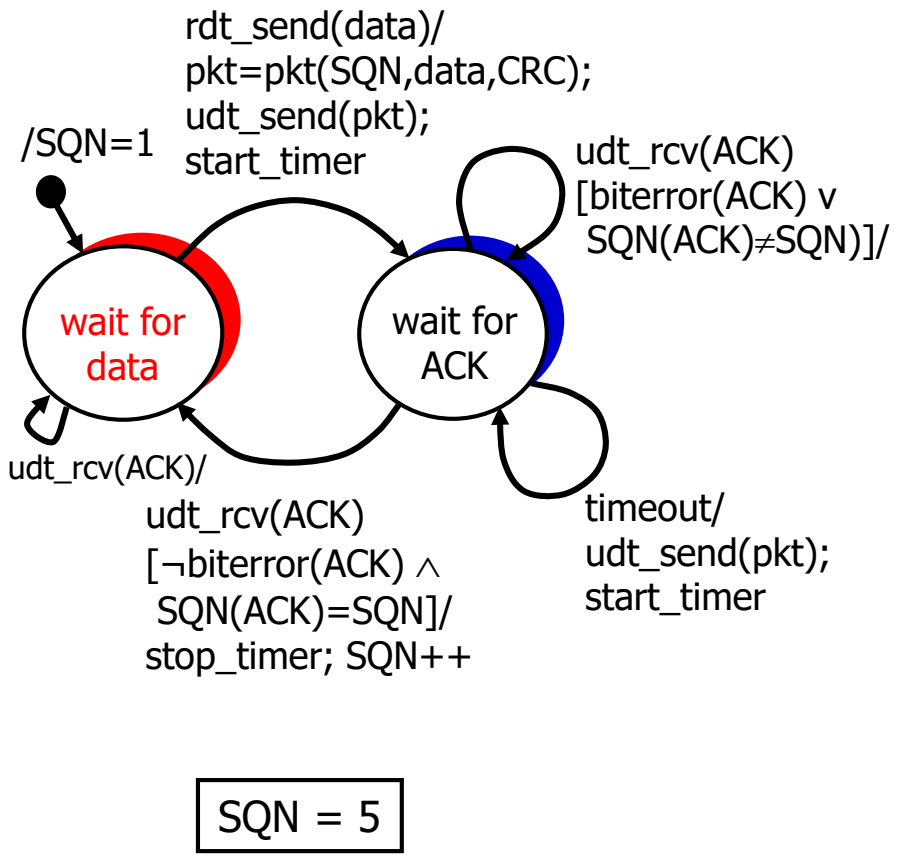


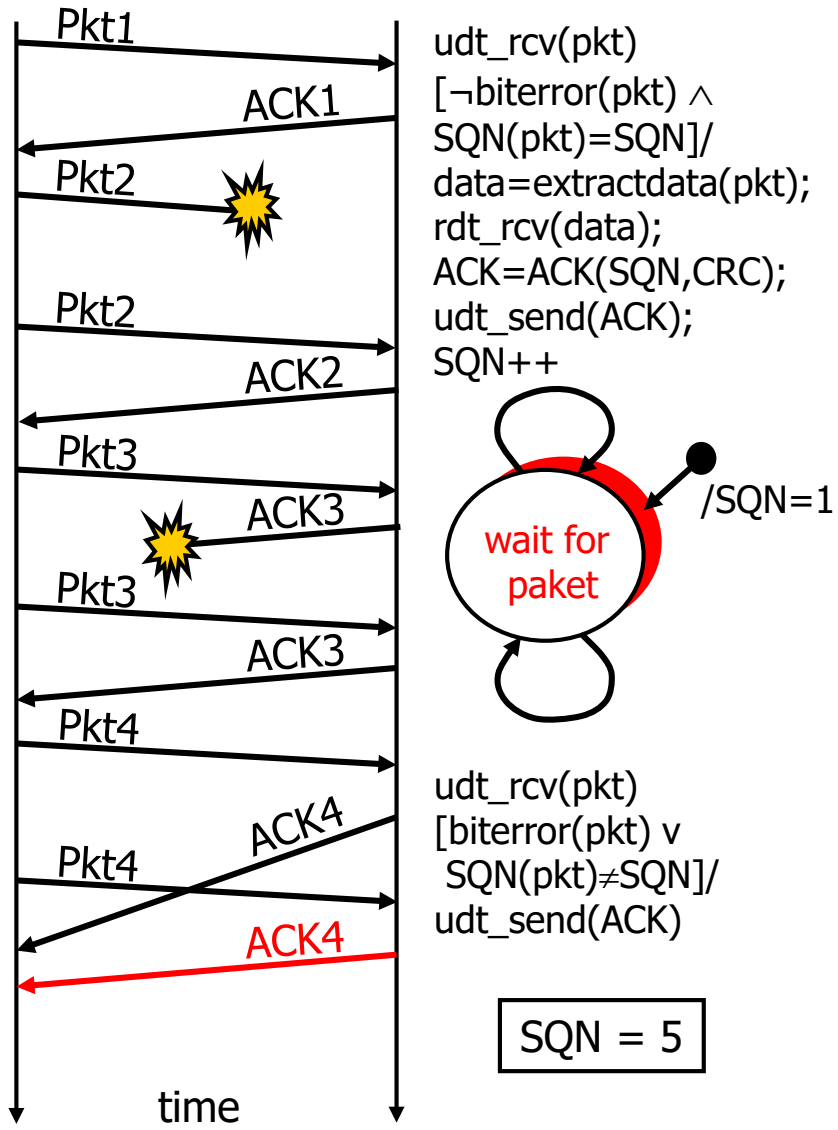
Stop-and-Wait: verzögertes ACK



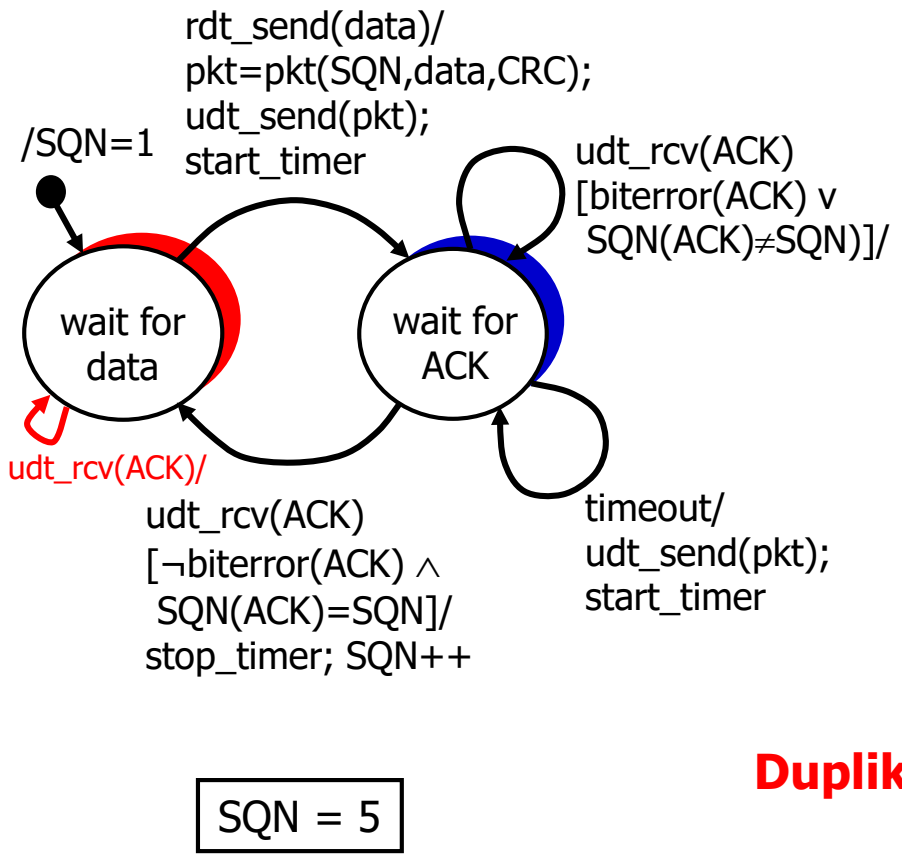


Stop-and-Wait: verzögertes ACK

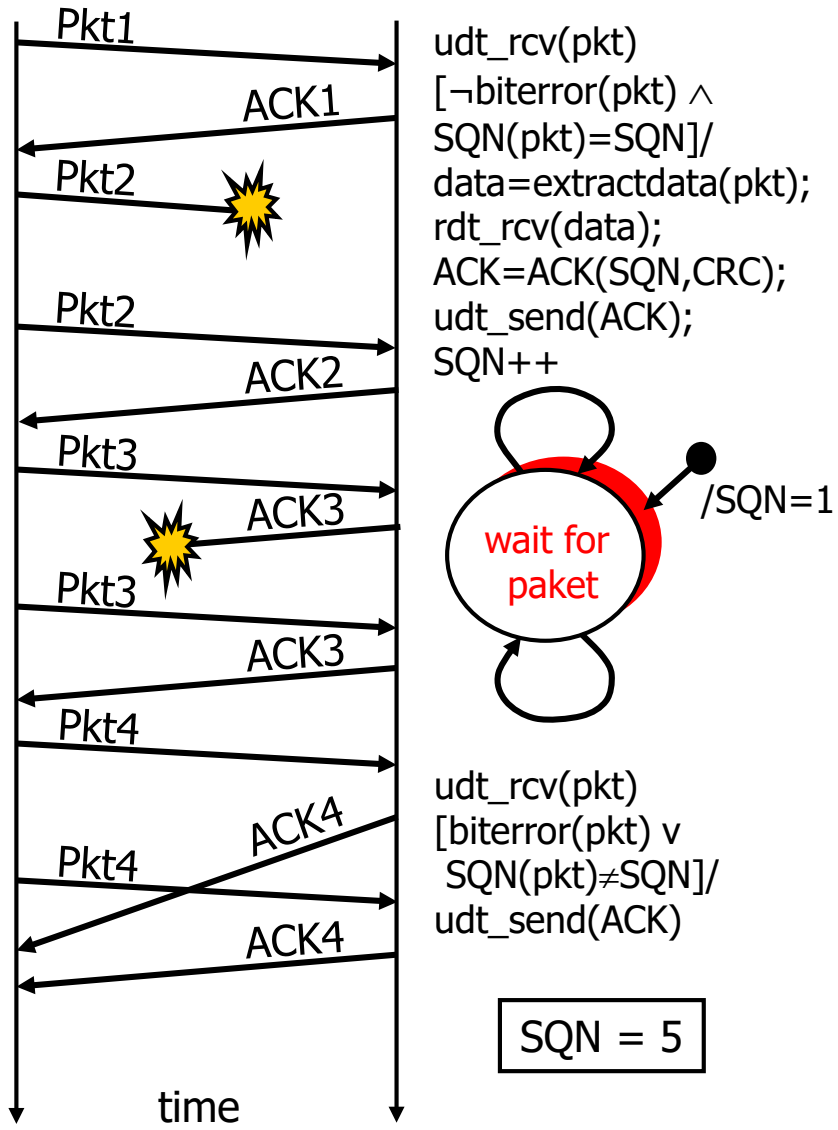




Stop-and-Wait: verzögertes ACK

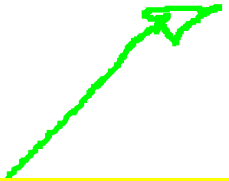


Duplikat!



Stop-and-Wait

- Sequenznummerraum
  - die Repräsentation der Sequenznummern ist endlich: ein Feld mit n Bits ermöglicht 2^n Sequenznummern
  - Wiederverwendung durch zyklisches Durchlaufen
  - für Stop-and-Wait ist ein Bit zur Darstellung von 2 Sequenznummern ausreichend: 0 und 1
  - Stop-and–Wait mit 0 und 1 als Sequenznummern heißt auch Alternating-Bit-Protokoll



man wartet ja auf erfolgreiche übertragung, also kann nie noch eine dritte paketnummer im umlauf sein

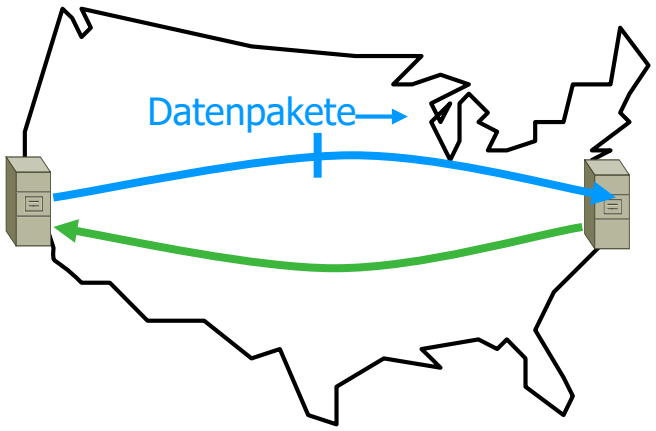
Transportschicht

- Einführung
- UDP
- Fehlerkontrolle
  - Stop-and-Wait
  - Go-Back-N und Selective Repeat
  - Leistungsanalyse
- TCP
- TLS
- QUIC

so ist leitung voll ausgelasted

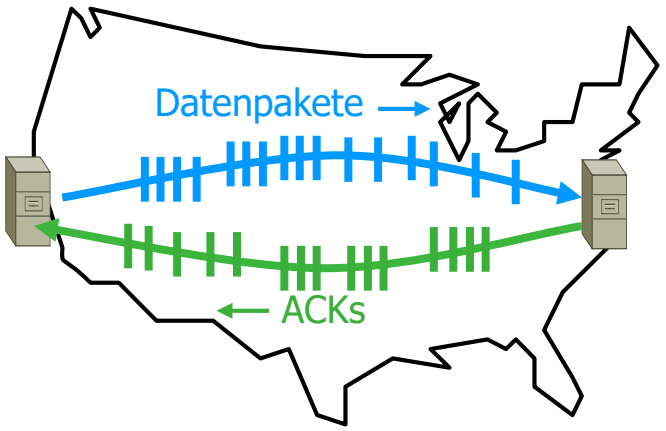
Go-Back-N und Selective Repeat

- Um die Ineffizienz von Stop-and-Wait zu vermeiden, senden Schiebefensterprotokolle mehrere Pakete, bevor die Bestätigung zurückkommt:



Stop-and-Wait

Meiste Zeit blockiert



Schiebefensterprotokoll

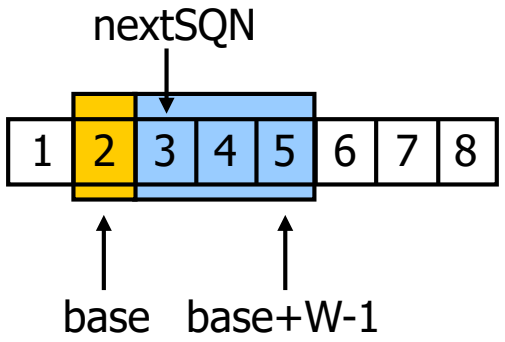
Quelle: Kurose, Ross. *Computer Networking: A Top-Down Approach*, 7th Ed., Pearson Education, 2017.

Go-Back-N

- Überblick über Go-Back-N
  - der Sender darf mehrere Pakete (bis zu einer Maximalzahl) vor Erhalt eines ACKs senden
  - er startet beim Senden des ersten Pakets einen Timer
  - er puffert die unbestätigten Pakete
  - wenn der Timer abläuft, werden alle unbestätigten Paket erneut gesendet
  - der Empfänger schickt kumulative ACKs: ein ACK mit einer SQN bedeutet, dass alle Pakete bis zu der SQN erfolgreich empfangen wurden wie kumulative wahrscheinlichkeit x<=SQN erhalten
  - der Empfänger akzeptiert nur Pakete in der richtigen Reihenfolge und benötigt keinen Puffer

Go-Back-N

- Sendepuffer



- base: SQN des ältesten unbestätigten Pakets
- nextSQN: SQN des nächsten zu verschickenden Pakets
- W: Fenstergröße, Anzahl der Pakete, die der Sender vor Erhalt eines ACKs senden darf
- das Fenster [base, base+W-1] wird beim Ablauf des Protokolls von links nach rechts verschoben, wegen der kumulativen ACKs hat es immer folgende Struktur:
  - [base, nextSQN-1]: versendete unbestätigte Pakete
  - [nextSQN, base+W-1]: bisher ungesendete Pakete, die vor Erhalt eines ACKs noch gesendet werden dürfen

Go-Back-N

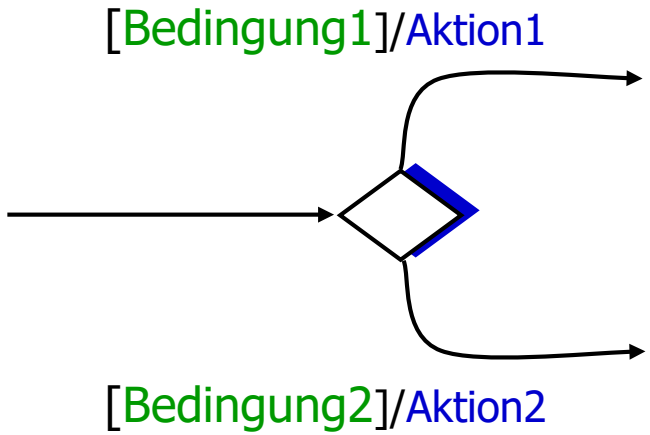
- informelle Beschreibung des Protokolls

  - Verhalten des Senders
    1. wenn Daten zum Senden und Platz im Fenster: sende Paket mit nextSQN und inkrementiere nextSQN; wenn es das erste Paket im Fenster ist, starte Timer
    2. wenn ein ACK ohne Bitfehler und mit SQN im Fenster zurückkommt, schiebe das Fenster bis zu dieser SQN; wenn das Fenster leer ist, stoppe den Timer, sonst starte den Timer neu
    3. wenn der Timeout abläuft, sende alle unbestätigten Pakete des Fensters erneut, starte den Timer erneut
  - Verhalten des Empfängers
    - wenn Paket ohne Bitfehler und mit aktueller SQN ankommt, sende ACK mit aktueller SQN und inkrementiere SQN, sonst sende das letzte ACK erneut (wie bei Stop-and-Wait)

Kein Buffer für out-of-order pakete notwendig (ist auf

Go-Back-N

- Beschreibung durch Statecharts
  - neues Element: Verzweigung



  - Zustand, in dem keine Zeit verbracht wird („Pseudozustand“)
  - abgehende Zustandsübergänge werden mittels Bedingungen gewählt, auslösende Ereignisse sind hier nicht möglich

Go-Back-N: Sender

cyclic redundancy check

OH-MY-FUCKING-GOD

barriere, sodass nicht m

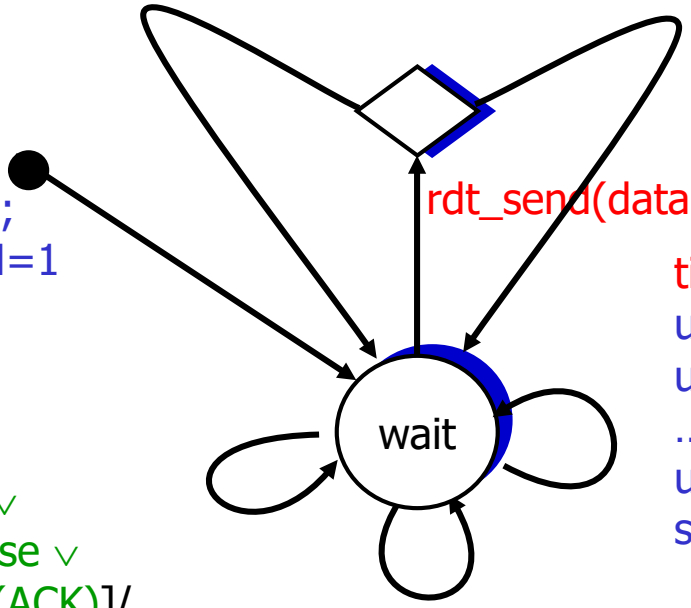
[nextSQN≥base+W]/
signal refusal to upper layer

[nextSQN<base+W]/
pkt[nextSQN]=pkt(nextSQN,data,CRC);
udt_send(pkt[nextSQN]);
if base=nextSQN start_timer;
nextSQN++

Senden von neuen daten, wenn Fer

Fenster Leer

/base=1;
nextSQN=1



timeout/
udt_send(pkt[base]);
udt_send(pkt[base+1]);
…
udt_send(pkt[nextSQN-1]);
start_timer

GO-BACK-N

ACK ist falsch->NOOP

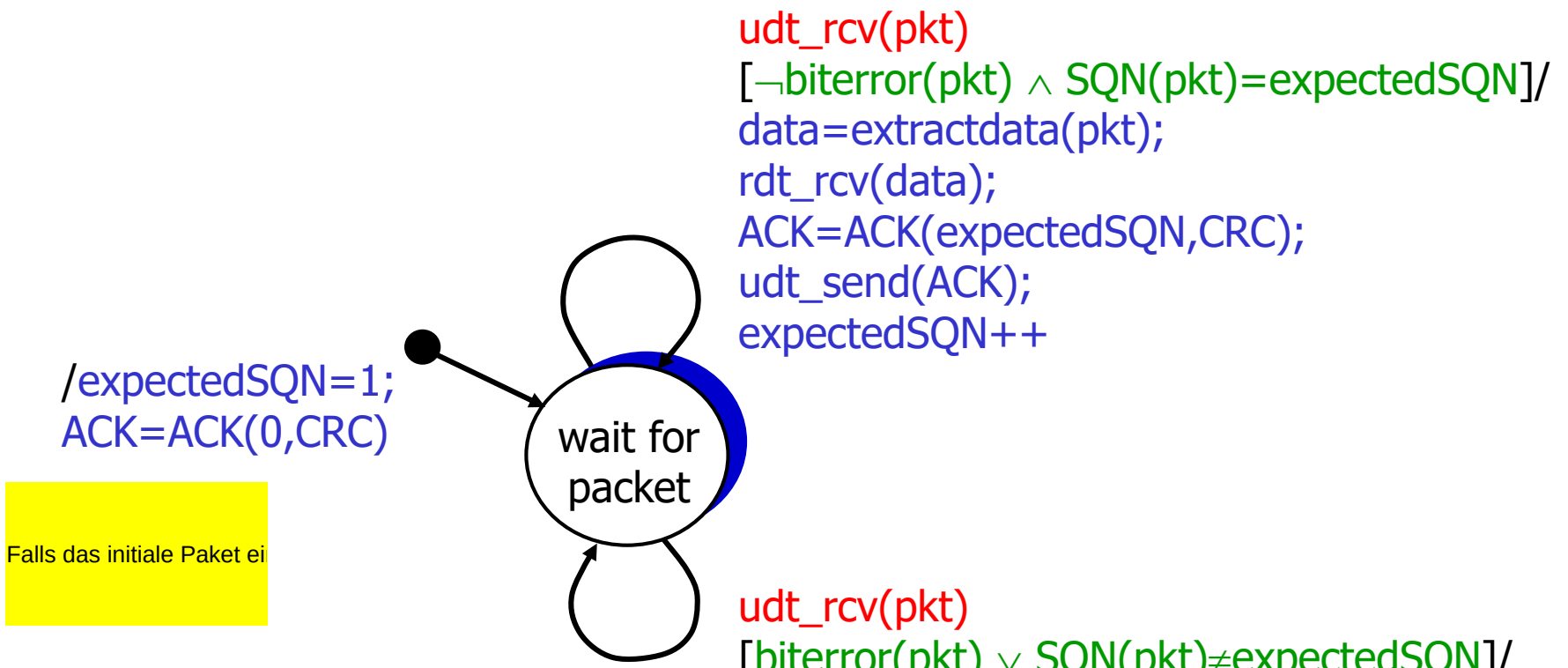
udt_rcv(ACK)
[biterror(ACK) ∨
SQN(ACK))<base ∨
nextSQN≤SQN(ACK)]/

udt_rcv(ACK)
[¬biterror(ACK) ∧ base≤SQN(ACK)<nextSQN]/
base=SQN(ACK)+1;
if base=nextSQN stop_timer else start_timer

Fenster leer, timer zurücksetzen

Quelle: Kurose, Ross. *Computer Networking: A Top-Down Approach*, 7th Ed., Pearson Education, 2017.

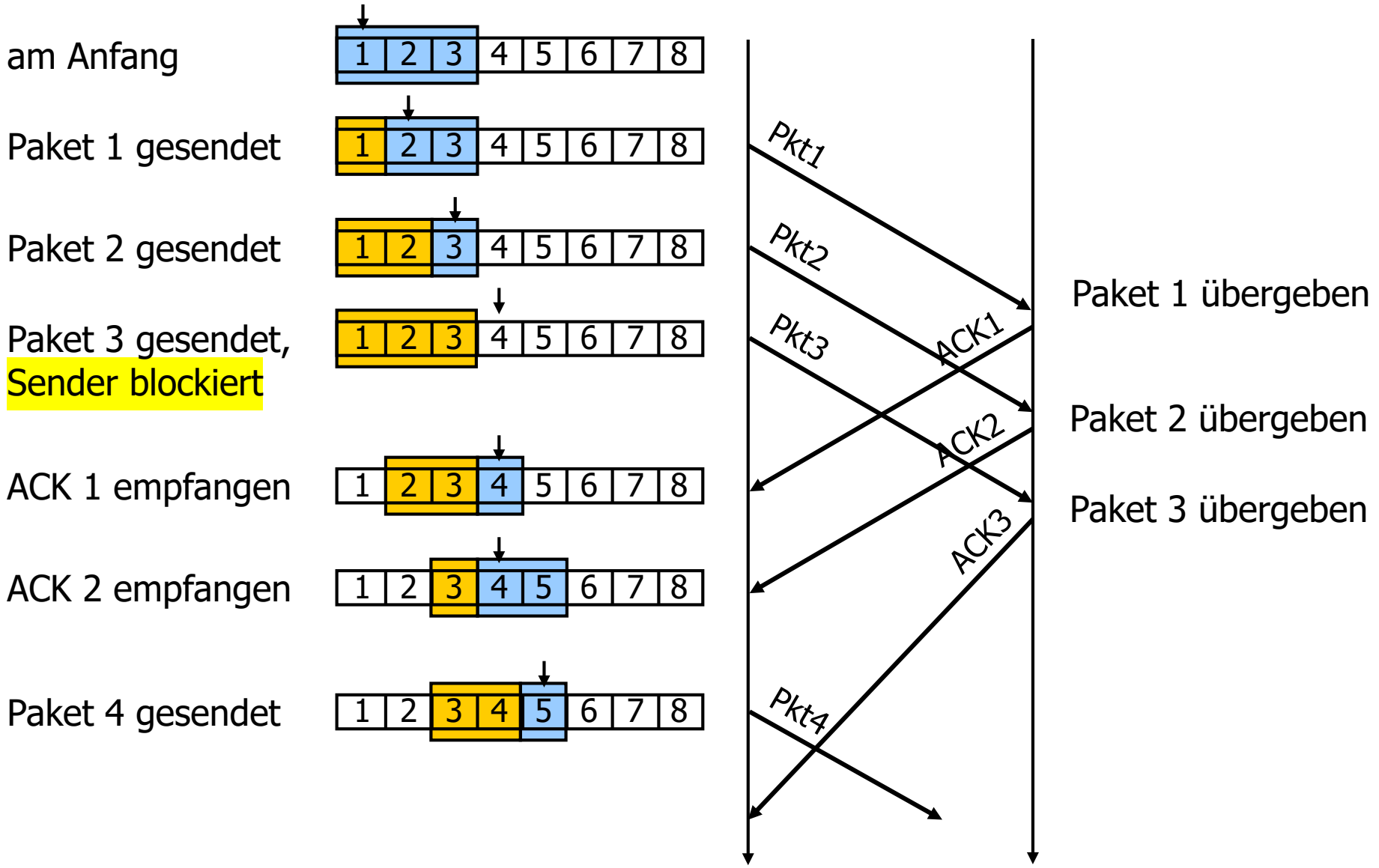
Go-Back-N: Empfänger



Falls das initiale Paket ei

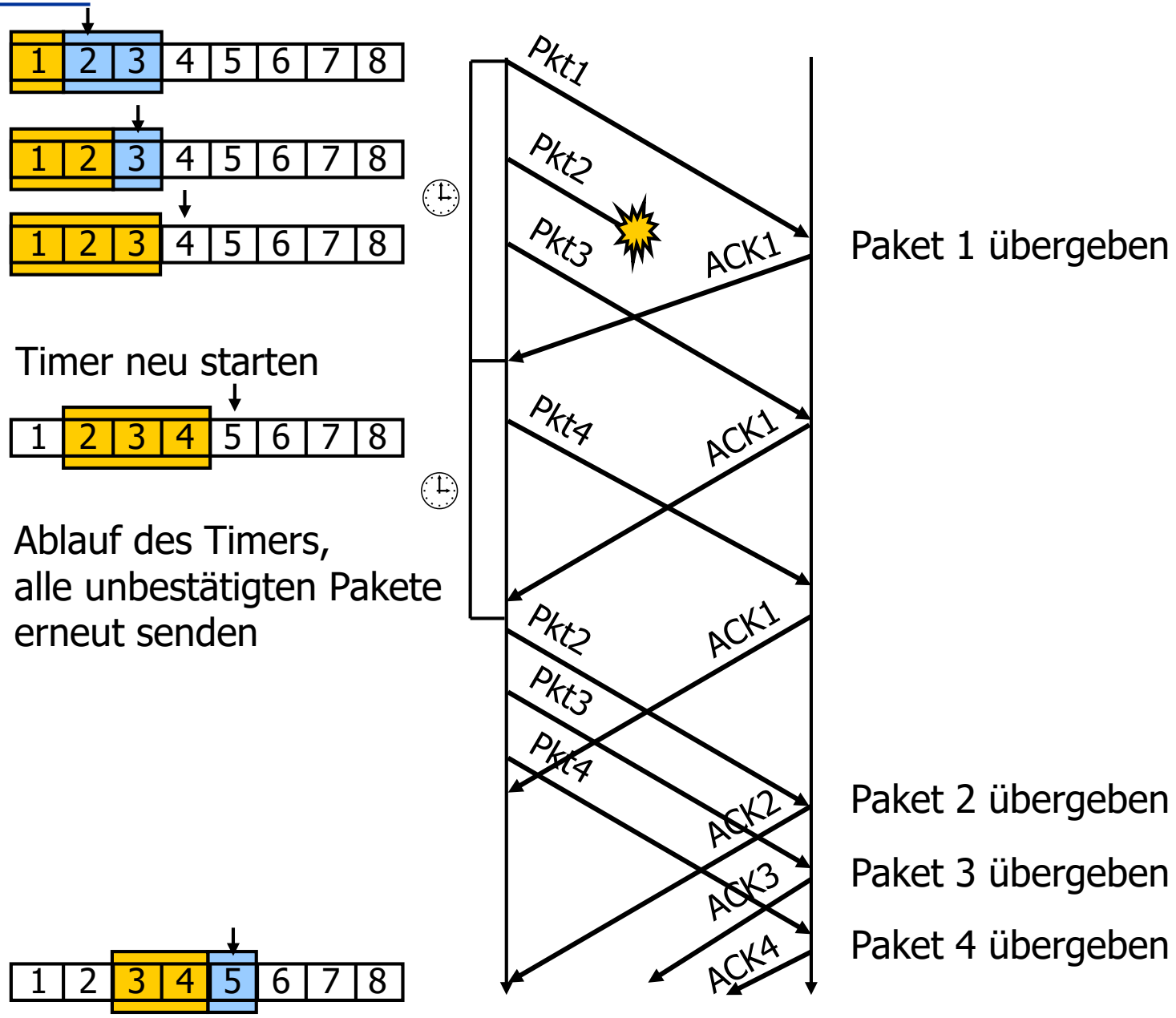
Quelle: Kurose, Ross. *Computer Networking: A Top-Down Approach*, 7th Ed., Pearson Education, 2017.

Go-Back-N: normaler Ablauf

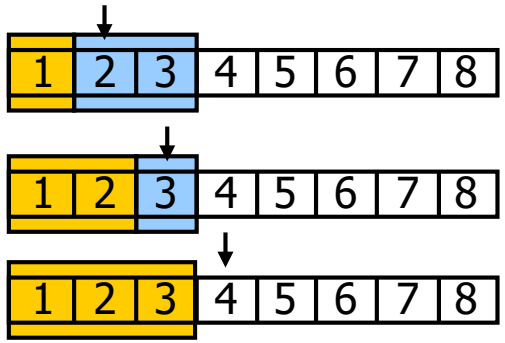


Fenster (eins) nach link

Go-Back-N: Paketverlust

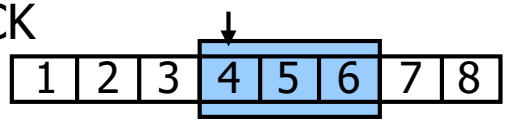


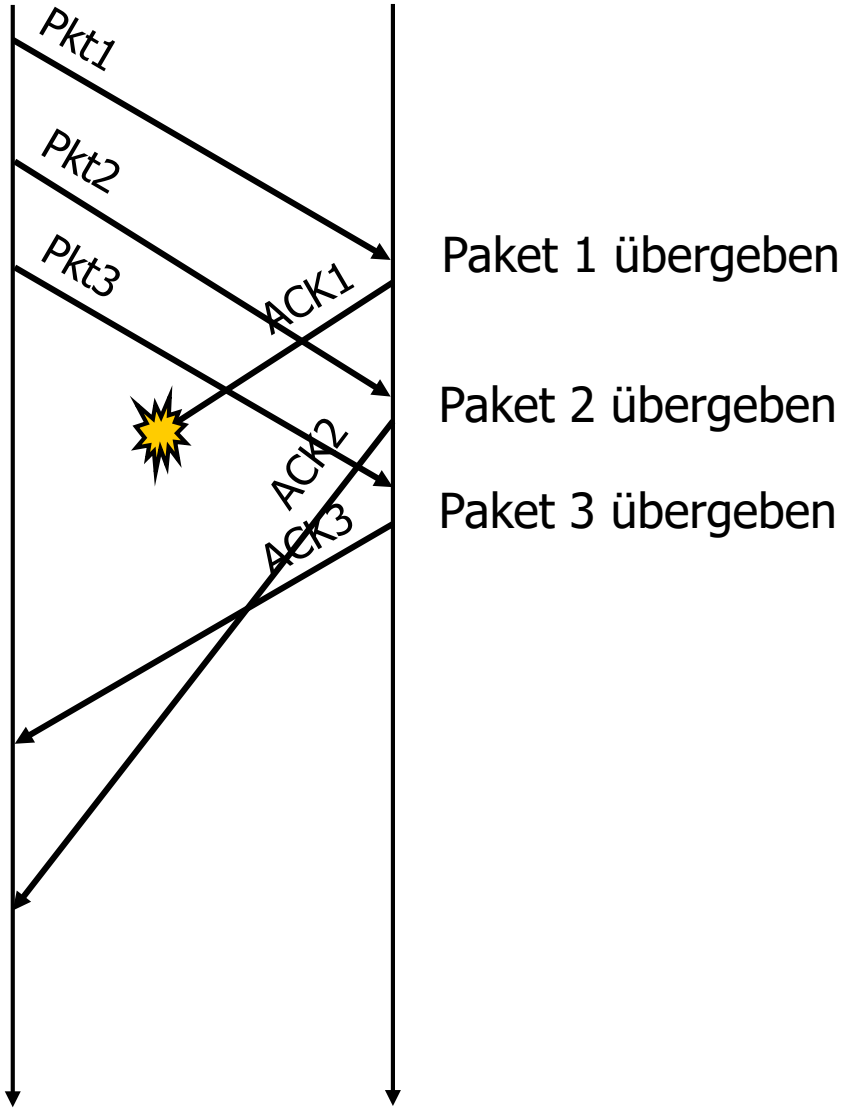
Go-Back-N: Verlust und Verspätung von ACKs



Vertauschungen/ Verlust eines ACKs tollerierbarer als bei Stop-and-wai

kumulatives ACK gleicht Verlust und Verspätung aus



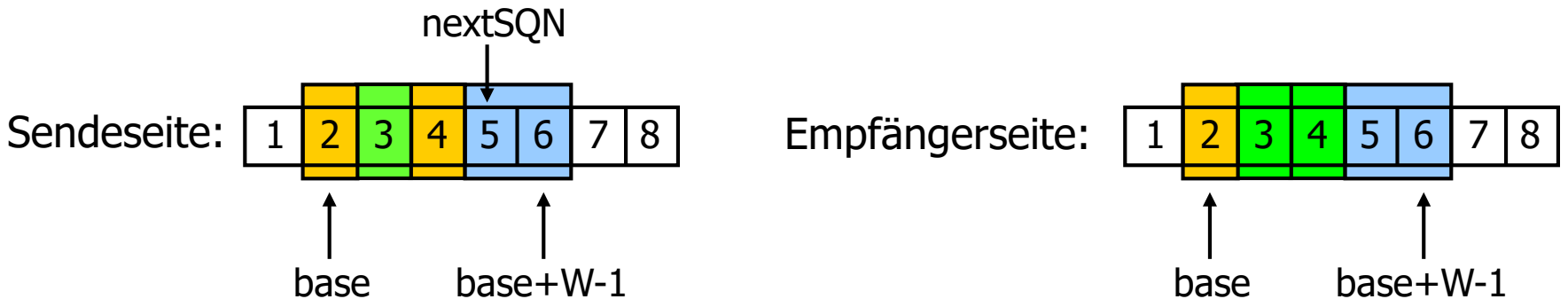


Selective Repeat

- Überblick über Selective Repeat
  - der Sender darf wieder mehrere Pakete (bis zu einer Maximalzahl) vor Erhalt eines ACKs senden
  - er startet beim Senden jedes Pakets einen Timer
  - er puffert die unbestätigten Pakete
  - wenn der Timer für ein Paket abläuft, wird dieses Paket erneut gesendet
  - der Empfänger schickt selektive ACKs: ein ACK mit einer SQN bedeutet nur, dass das Paket mit der SQN erfolgreich empfangen wurde
  - der Empfänger benötigt einen Puffer zum Ausgleich von Lücken beim Empfang

während go-back-N auch mit stop-and-wait client funktioniert, funktioniert Selective repeate nur mit änderungen am client

Selective Repeat: Sende- und Empfängerpuffer



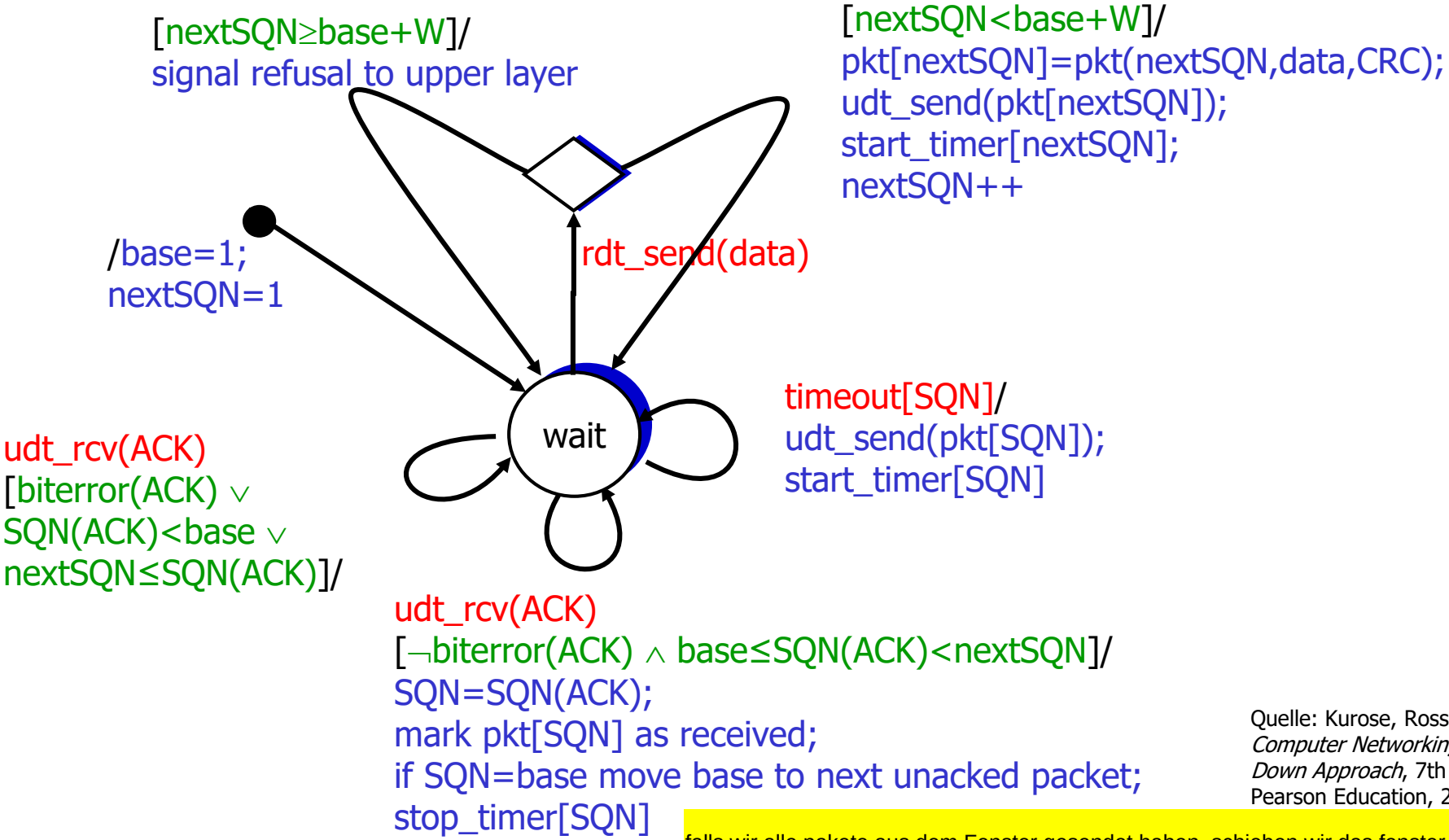
- base, nextSQN, W: wie bei Go-Back-N
- das Fenster auf Sendeseite enthält versendete unbestätigte, versendete bestätigte und ungesendete Pakete
- der Empfänger puffert die empfangenen Pakete
- das Fenster auf Empfängerseite enthält empfangene Pakete und Lücken und Platz für unempfangene Pakete

Selective Repeat

- informelle Beschreibung des Protokolls

  - Verhalten des Senders
    1. wenn Daten zum Senden und Platz im Fenster: sende Paket, starte Timer für dieses Paket und inkrementiere nextSQN
    2. wenn ein ACK ohne Bitfehler und mit SQN im Fenster zurückkommt, markiere das Paket mit SQN als bestätigt, schiebe das Fenster bis zur nächsten Lücke
    3. wenn der Timeout für das Paket mit SQN abläuft, sende dieses Paket erneut, starte den Timer für dieses Paket erneut
  - Verhalten des Empfängers
    - wenn Paket ohne Bitfehler und mit SQN im Fenster ankommt, sende ACK mit dieser SQN, puffere das Paket und schiebe das Fenster bis zur nächsten Lücke
    - wenn Paket mit SQN aus vorigem Fenster ankommt, sende das ACK hierfür erneut

Selective Repeat: Sender

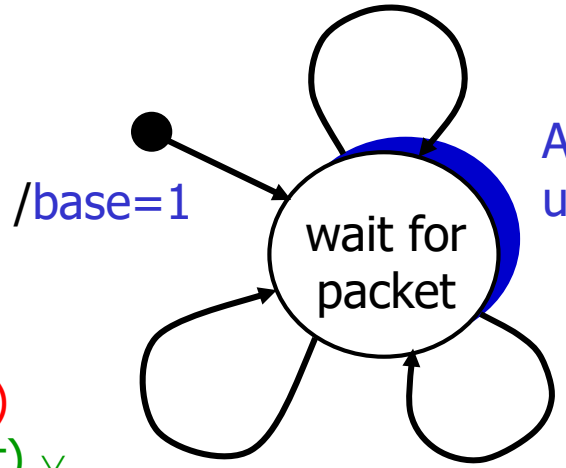


Quelle: Kurose, Ross. *Computer Networking: A Top-Down Approach*, 7th Ed., Pearson Education, 2017.

falls wir alle pakete aus dem Fenster gesendet haben, schieben wir das fenster so lange weiter, bis

Selective Repeat: Empfänger

udt_rcv(pkt)
[¬biterror(pkt) ∧ SQN(pkt) in [base,base+W-1]]/
data=extractdata(pkt);
buffer data;
if SQN(pkt)=base
   base=lowest unrcvd sequence number;
   deliver all data until base-1;
ACK=ACK(SQN(pkt),CRC);
udt_send(ACK)



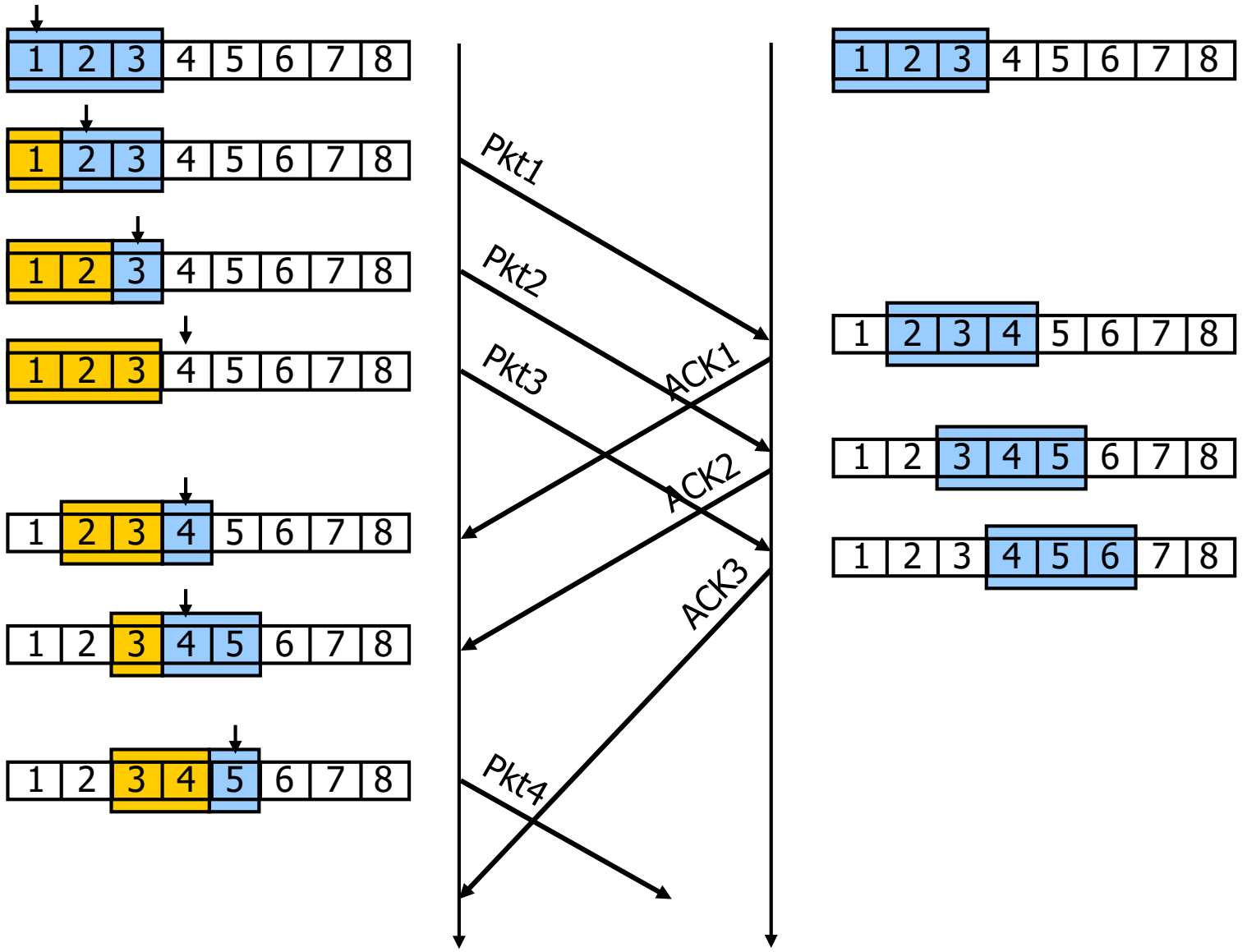
udt_rcv(pkt)
[biterror(pkt) ∨
SQN(pkt) not in [base-W,base+W-1]]/

udt_rcv(pkt)
[¬biterror(pkt) ∧ SQN(pkt) in [base-W,base-1]]/
ACK=ACK(SQN(pkt),CRC);
udt_send(ACK)

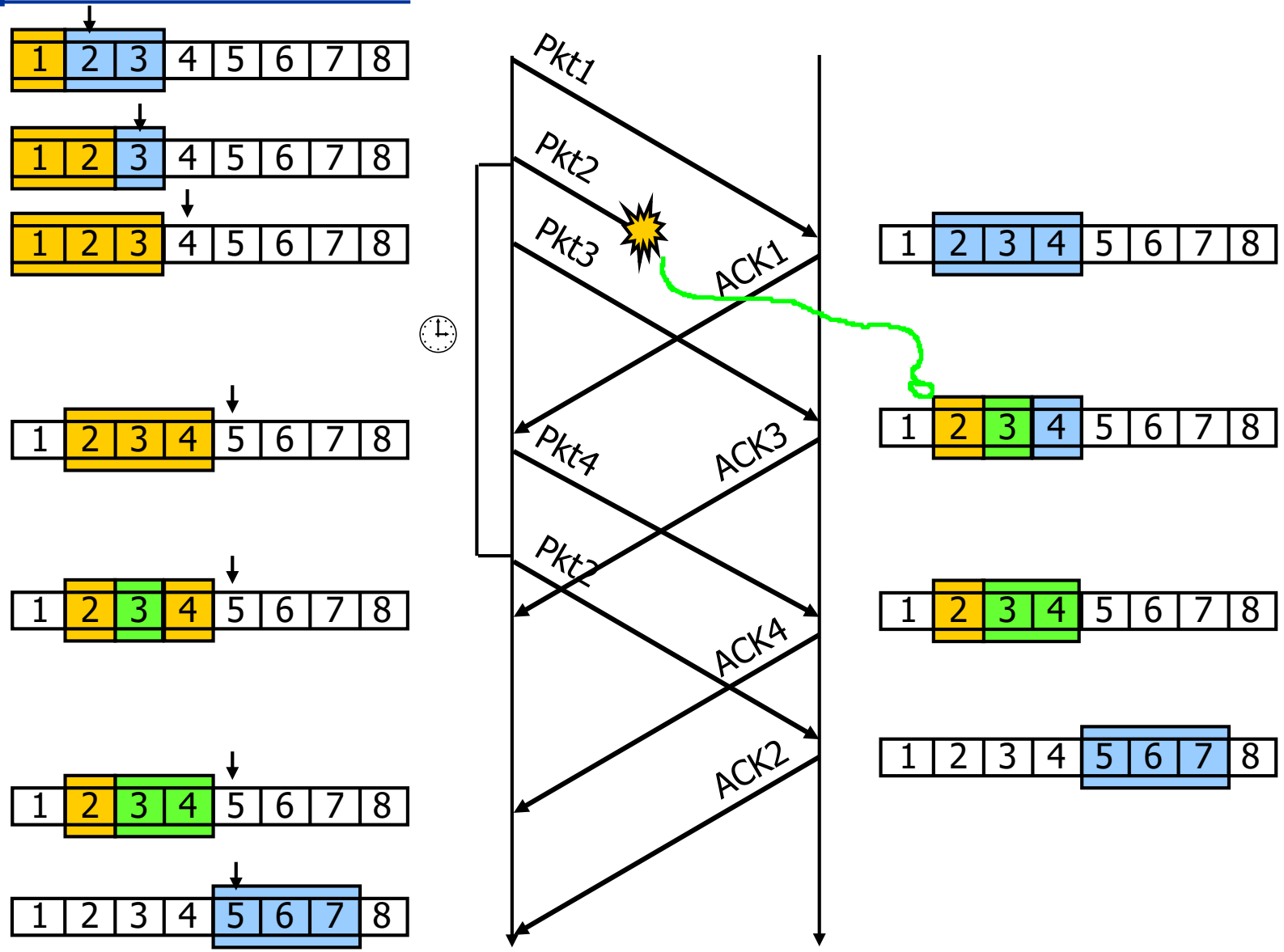
Quelle: Kurose, Ross,
wenn ein paket ein zweites mal wieder ankommt (weil das erste ACK verloren
Pearson Education, 2017.

Selective Repeat: normaler Ablauf

gelb heißt verschickt, aber noch ke

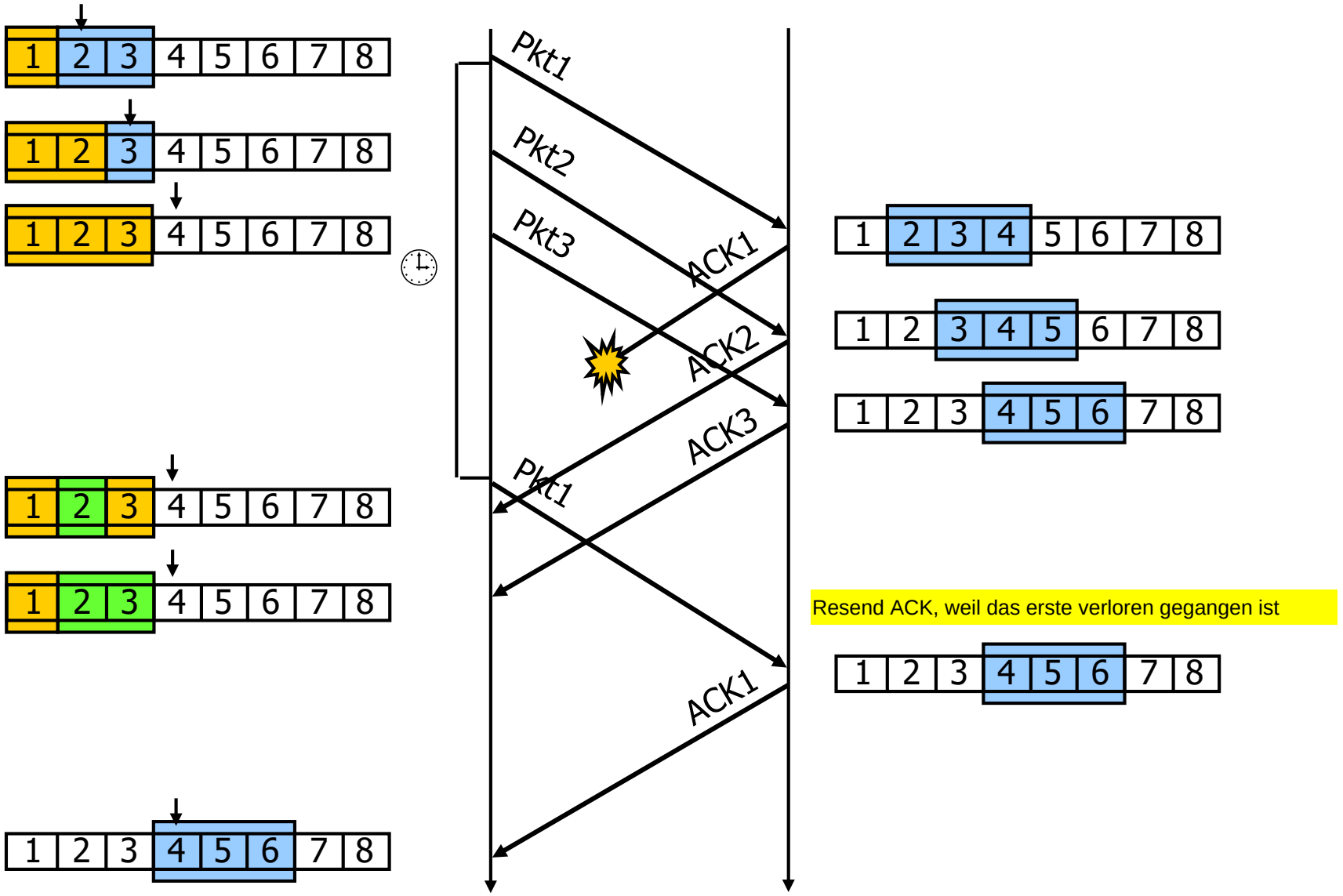


Selective Repeat: Paketverlust



grün, ACK erhalten, aber mitte

Selective Repeat: Verlust eines ACKs



Selective Repeat

- Sequenznummerraum bei Schiebefensterprotokollen
  - endliches Sequenznummerfeld mit m Werten
  - zyklisches Durchlaufen: Wiederverwendung von SQN
  - unterschiedliche Pakete mit gleicher SQN müssen unterschieden werden
  - hinreichende Bedingungen dafür:
    - falls Empfangsfenstergröße = 1: W < m
    - falls Sendefenstergröße = Empfangsfenstergröße = W > 1: W < (m+1)/2

M ist die anzahl der Sequenznummern und W die größe des schiebefensters

- Beispiel für zu kleinen Sequenznummerraum
  - m = 4 Sequenznummern, Fenstergröße W = 3
  - W > (m+1)/2
  - Empfänger kann nicht unterscheiden, ob Paket 0 alt oder neu ist, siehe nächste Seite

umstellen: 2W-1<m, also m muss mindestens doppelt so groß

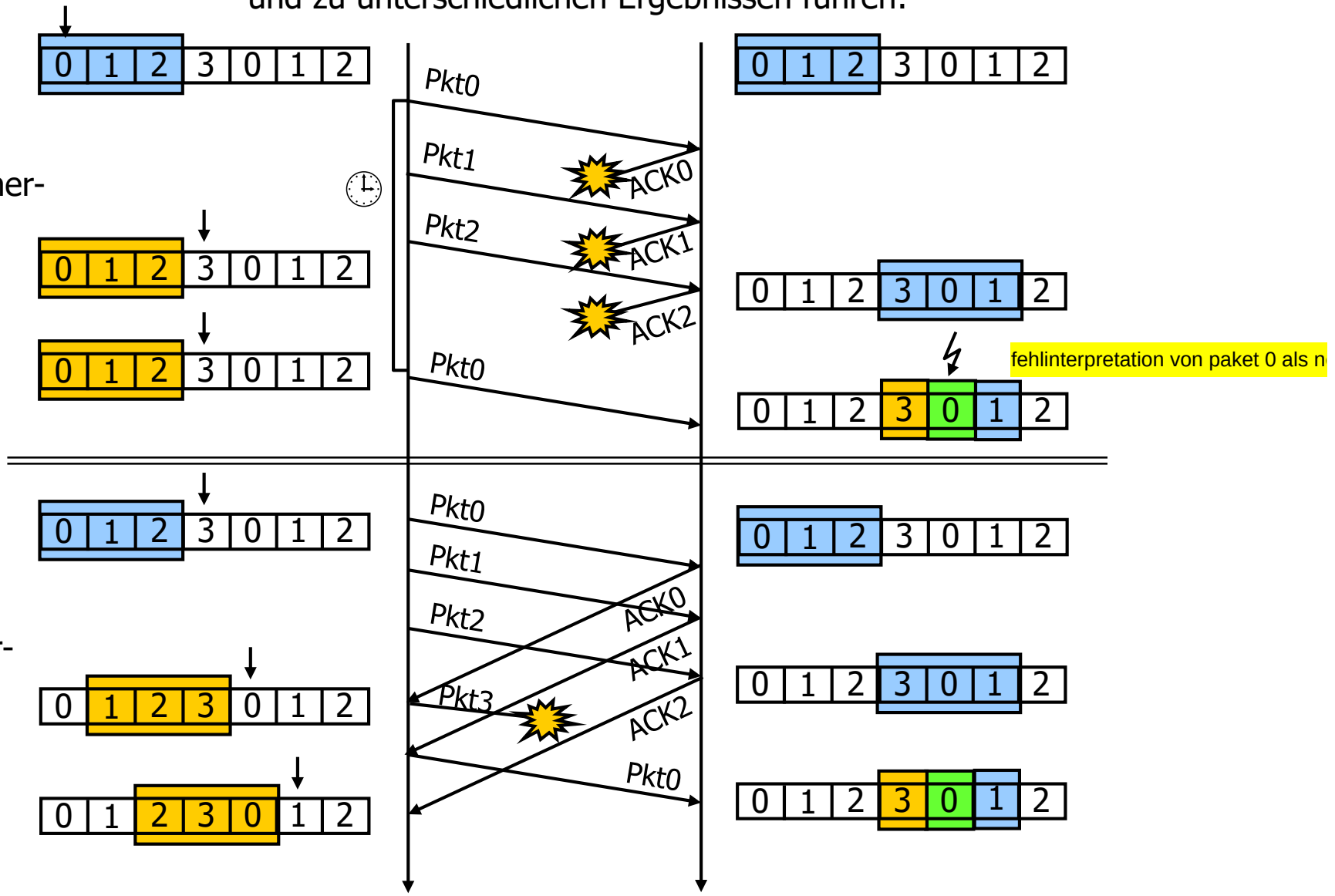
hier geht es schief 3> (4+1)/2 = 2.5

Selective Repeat

zwei mögliche Abläufe für m=4 und W=3,
die für den Empfänger nicht unterscheidbar sind
und zu unterschiedlichen Ergebnissen führen:

hier wird altes Paket 0 fälschlicherweise für neu gehalten

hier wird neues Paket 0 korrekterweise für neu gehalten



Vergleich von Go-Back-N und Selective Repeat

- Vorteile Go-Back-N
  - kumulative ACKs gleichen ACK-Verluste und -Verspätungen schnell aus, ohne dass die Pakete erneut gesendet werden müssen
  - der Sender benötigt nur einen Timer
  - der Empfänger benötigt keinen Puffer
  - Sender und Empfänger können einfacher realisiert werden, weil keine Lücken in den Fenstern beachtet werden müssen
- Vorteil Selective Repeat
  - weniger Wiederholungen von Sendungen, weil nur wirklich fehlerhafte oder verlorengegangene Pakete erneut gesendet werden

dafür ist die wiederholung teurer, weil das paket evtl. mehrmals gesendet werd

Transportschicht

- Einführung
- UDP
- Fehlerkontrolle
  - Stop-and-Wait
  - Go-Back-N und Selective Repeat
  - Leistungsanalyse
- TCP
- TLS
- QUIC

Leistungsanalyse

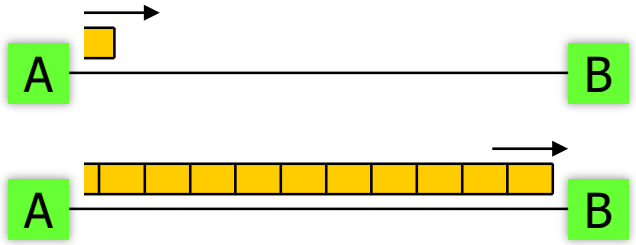
- Fragen
  - wann tritt bei Stop-and-Wait eine Senderblockade ein und wie stark wird der mögliche Durchsatz verkleinert?
  - wie groß muss bei Schiebefensterprotokollen das Fenster sein, um den Kanal zu füllen?
  - ist Go-Back-N oder Selective Repeat effizienter?
- im folgenden analytische Betrachtung aus
  - W. Stallings: *Data and Computer Communications*, 9th Ed., Pearson Education, 2013
  - Beispiel für typische Leistungsanalyse von Kommunikationssystemen
  - einige vereinfachende Annahmen sind nötig, um rechnen zu können
  - die mathematischen Ausdrücke sind gar nicht so „schlimm"
  - bei Go-Back-N benötigen wir die meisten Vereinfachungen und schwierigsten Ausdrücke
  - noch genauere Untersuchungen sind mit Simulation möglich

Leistungsanalyse

- Produkt aus Bitrate und Verzögerung
  - Bitrate R, Ausbreitungsverzögerung D vom Sender zum Empfänger
  - einfacher Kanal, A sendet ohne Unterbrechung an B

R wie schnell können pakete auf d

RD > 1:

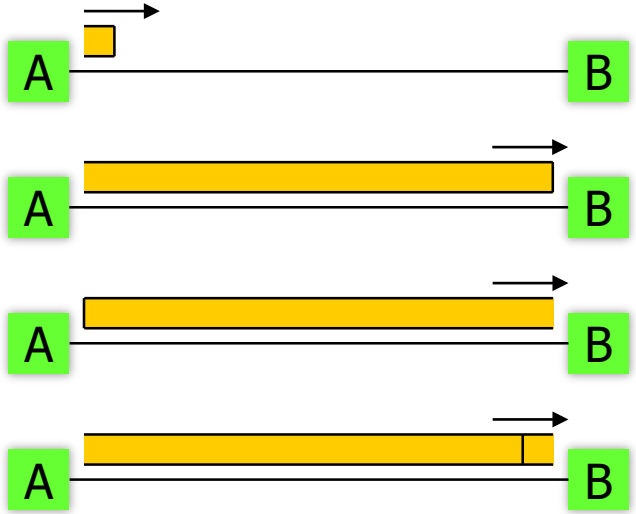


Normalfall

t = 0: A beginnt zu senden

t = D: erstes Bit erreicht B,
RD Bits sind mittlerweile gesendet

RD < 1:



t = 0: A beginnt zu senden

t = D: der Anfang des Bits erreicht B,
RD·100% des Bits sind mittlerweile gesendet

Erst Teil des Pakets gesendet

t = 1/R: das Ende des Bits verlässt A

t = 1/R + D: das Ende des Bits erreicht B

Quelle: Stallings: *Computer Networking with Internet Protocols and Technology*, Pearson Education, 2004.

Leistungsanalyse

- Kanalpuffergröße in Bits

$$R \cdot D = \frac{D}{1/R} = \frac{l/v}{1/R} = \frac{\text{Ausbreitungsverzögerung}}{\text{Bitsendezeit}}$$

= Anzahl gesendeter Bits während sich das erste Bit vom Sender zum Empfänger ausbreitet = Kanalpuffergröße in Bits

- Beispiel für RD > 1:
  - R = 100 Mbps, l = 4800 km, v = $3 \cdot 10^8$ m/s
  - $RD = 100 \cdot 10^6 \frac{\text{Bits}}{\text{s}} \cdot \frac{4800 \cdot 10^3 \text{m}}{3 \cdot 10^8 \text{m/s}} = 1600 \cdot 10^3 \text{Bits} \approx 195 \text{KB}$
- Beispiel für RD < 1:
  - R = 10 Mbps, d = 10 m, v = $2 \cdot 10^8$ m/s
  - $RD = 10 \cdot 10^6 \frac{\text{Bits}}{\text{s}} \cdot \frac{10 \text{m}}{2 \cdot 10^8 \text{m/s}} = 0{,}5 \text{Bits}$

Leistungsanalyse

Kanalpuffergröße in Paketen

- mit Paketgröße L:

$$a = \frac{R \cdot D}{L} = \frac{l / v}{L / R} = \frac{Ausbreitungsverzögerung}{Paketsendezeit}$$

= Anzahl gesendeter Pakete während sich das erste Bit vom Sender zum Empfänger ausbreitet = Kanalpuffergröße in Paketen

Leistungsanalyse

- Beispiel:
  L = 1500 Byte, v = $3 \cdot 10^8$ m/s
  - Glasfaser: R = 100 Mbit/s, l = 10 km
    - Ausbreitungsverzögerung $l/v \approx 33\ \mu s$
    - Kanalpuffergröße $a = \frac{l/v}{L/R} \approx 0{,}28$
  - Satellitenkommunikation: R = 36 Mbit/s, l = 72.000 km
    - Ausbreitungsverzögerung $l/v = 240.000\ \mu s$
    - Kanalpuffergröße $a = \frac{l/v}{L/R} = 720$

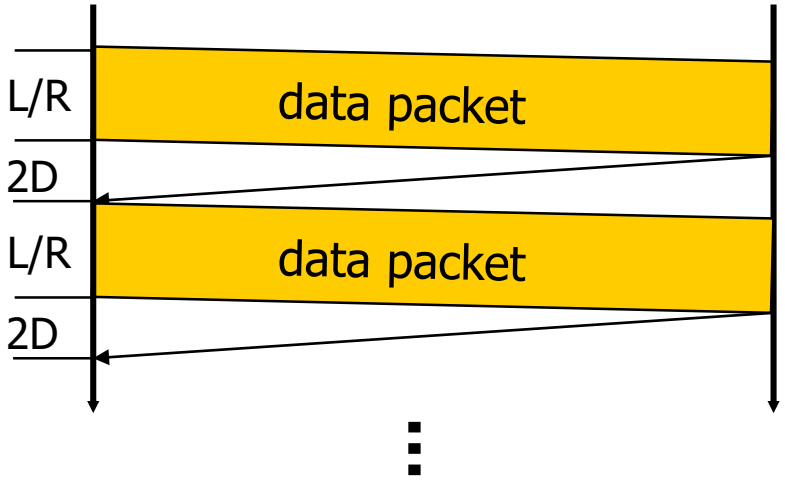
- einige Werte für a ($v = 3 \cdot 10^8$ m/s):

Bitrate	Paketgröße	Entfernung	a
500 Kbps	136 Bits	10 m	0,0001
1 Mbps	1500 Bytes	10 m	0,0000028
1 Mbps	1500 Bytes	1 Km	0,00028
1 Mbps	1500 Bytes	10 Km	0,0028
1 Mbps	1500 Bytes	100 Km	0,028
1 Mbps	1500 Bytes	1.000 Km	0,28
1 Mbps	1500 Bytes	10.000 Km	2,8
1 Mbps	1500 Bytes	36.000 Km	10
10 Mbps	1500 Bytes	10 Km	0,028
10 Mbps	1500 Bytes	100 Km	0,28
100 Mbps	1500 Bytes	100 m	0,0028
100 Mbps	1500 Bytes	10 km	0,28
100 Mbps	1500 Bytes	1.000 km	27,8
1 Gbps	1500 Bytes	100 m	0,028
1 Gbps	1500 Bytes	10 km	2,8
1 Gbps	1500 Bytes	1.000 km	277,8
1 Gbps	1500 Bytes	36.000 km	10.000
100 Gbps	1500 Bytes	100 m	2,8
100 Gbps	1500 Bytes	10 km	277,8
100 Gbps	1500 Bytes	1.000 km	27.777,8
100 Gbps	1500 Bytes	36.000 km	1.000.000

Quelle: Stallings: *Computer Networking with Internet Protocols and Technology*, Pearson Education, 2004.

Leistungsanalyse: Stop-and-Wait

- Stop-and-Wait ohne Fehler
  - Vernachlässigung der ACK-Sendezeit und Bearbeitungszeiten (sinnvolle vereinfachende Annahme für diese Berechnungen)



  - pro Zeit gesendete Bits:

$$\text{Durchsatz} = \frac{L}{L/R + 2D}$$

2D = Call + ResponseL/R = Zeit die ein pa

  - normiert durch die Bitrate (gut für Vergleich bei verschiedenen Bitraten, vergleiche auch mit vorletzter Folie):

$$\text{normierter Durchsatz} = S = \frac{L}{L/R + 2D} \cdot \frac{1}{R} = \frac{1}{1 + 2RD/L} = \frac{1}{1 + 2a}$$

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

$\Rightarrow$ schlechter Durchsatz für große a (Kanal kann nicht gefüllt werden)

Wenn a größer wird, muss der sender immer länger warten

Leistungsanalyse Stop-and-Wait

- Stop-and-Wait mit Fehlern
  - Sendewiederholung nach einem Fehler (Timeout oder fehlerhaftes ACK)
  - Annahme: Fehler treten unabhängig voneinander mit Wahrscheinlichkeit p auf (schon wieder eine Vereinfachung!)
  - Timeout = 2D auch vereinfachung, weil D nicht konstant, muss es eher max(D_t) sein
  - N ist die mittlere Anzahl, mit der jedes Paket gesendet werden muss, dann:

    wie oft muss eine Sendung wiederholt werden

$$\text{Durchsatz} = \frac{L}{N \cdot (L/R + 2D)}$$

$$S = \frac{L/R}{N \cdot (L/R + 2D)} = \frac{1}{N \cdot (1 + 2RD/L)} = \frac{1}{N \cdot (1 + 2a)}$$

Leistungsanalyse: Stop-and-Wait

- Berechnung von N:
  - die Wahrscheinlichkeit, dass ein Paket i-mal gesendet werden muss, ist gleich der Wahrscheinlichkeit von i-1 fehlerhaften Sendungen gefolgt von einem fehlerfreien Senden
  - Pr[i Sendeversuche] = $p^{i-1} \cdot (1-p)$
  - dies ist die geometrische Verteilung, Erwartungswert:

$$N = E[\text{Sendeversuche}] = \sum_{i=1}^{\infty} i \cdot Pr[i \text{ Sendeversuche}] =$$

$$\sum_{i=1}^{\infty} i \cdot p^{i-1} \cdot (1-p) = (1-p) \sum_{i=1}^{\infty} i \cdot p^{i-1} = (1-p) \left(\sum_{i=0}^{\infty} p^i \right)' = (1-p) \left(\frac{1}{1-p} \right)' =$$

$$(1-p) \frac{1}{(1-p)^2} = \frac{1}{1-p}$$

1/N = p-1

  - Einsetzen liefert den normalisierten Durchsatz:

$$S = \frac{1-p}{1+2a}$$

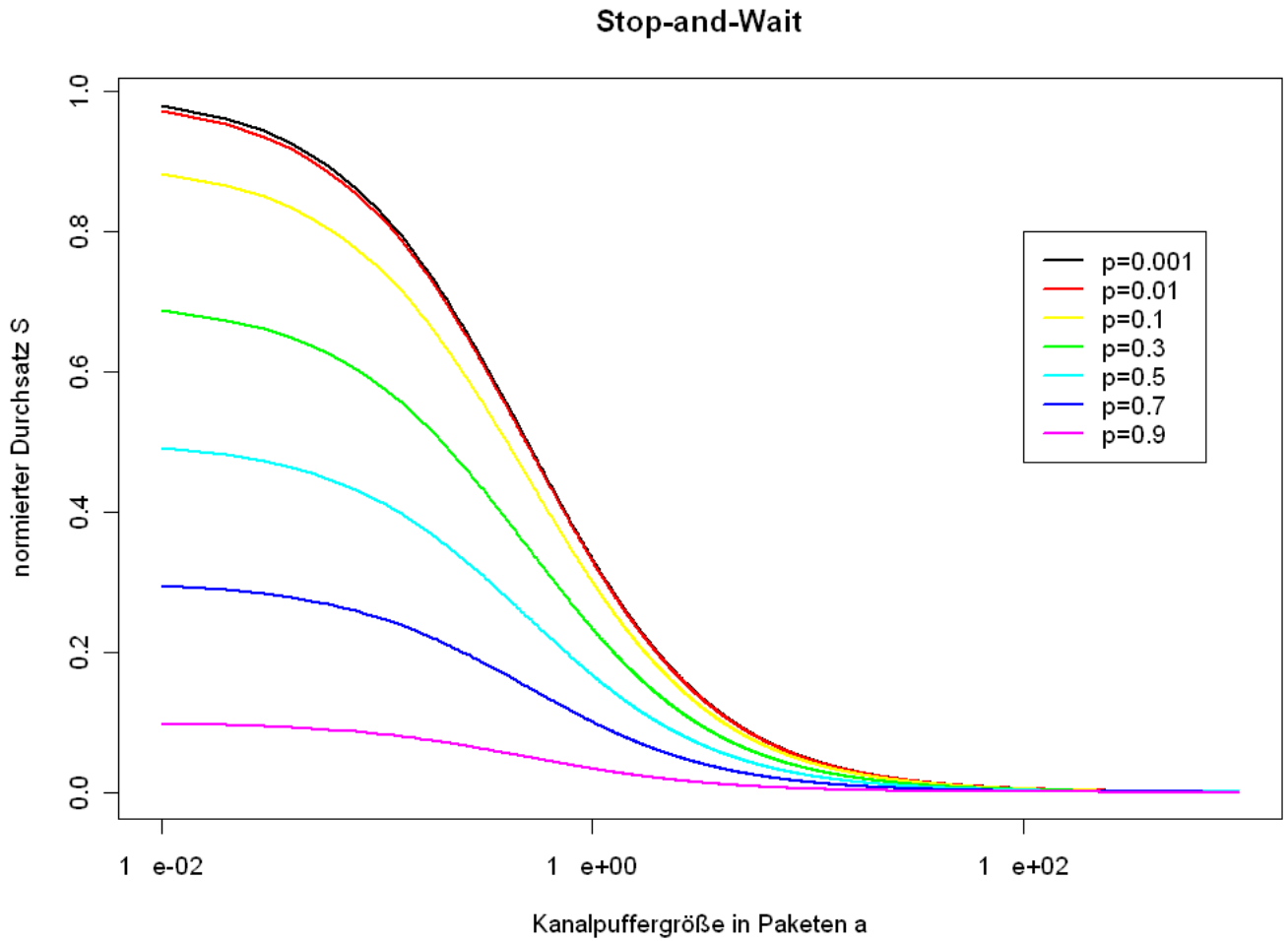
Fehlerbereinigt

$\Rightarrow$ schlechter Durchsatz für große a und p

höhere fehlerwahrscheinlichkeit, kleinerer Durchsatz

Leistungsanalyse: Stop-and-Wait

- normierter Durchsatz von Stop-and-Wait als Funktion von a:



für große a fällt der Durchsatz ab, für größere p ist der Durchsatz ebenfalls kleiner

a ist der dominanteste parameter für Stop-and-wait

Leistungsanalyse: Schiebefensterprotokolle

- Schiebefensterprotokolle ohne Fehler
  - für Fenstergröße mit W Paketen der Länge L
  - Fall 1: das Fenster ist groß genug, um zu senden, bis ACK zurückkommt:
    - $W \geq \frac{L/R + 2D}{L/R} = 1 + 2a$
    - $S = \frac{W \cdot L}{W \cdot L/R} \cdot \frac{1}{R} = 1$

Ein paket senden und AC

anzahl der bits die pro fenster ges

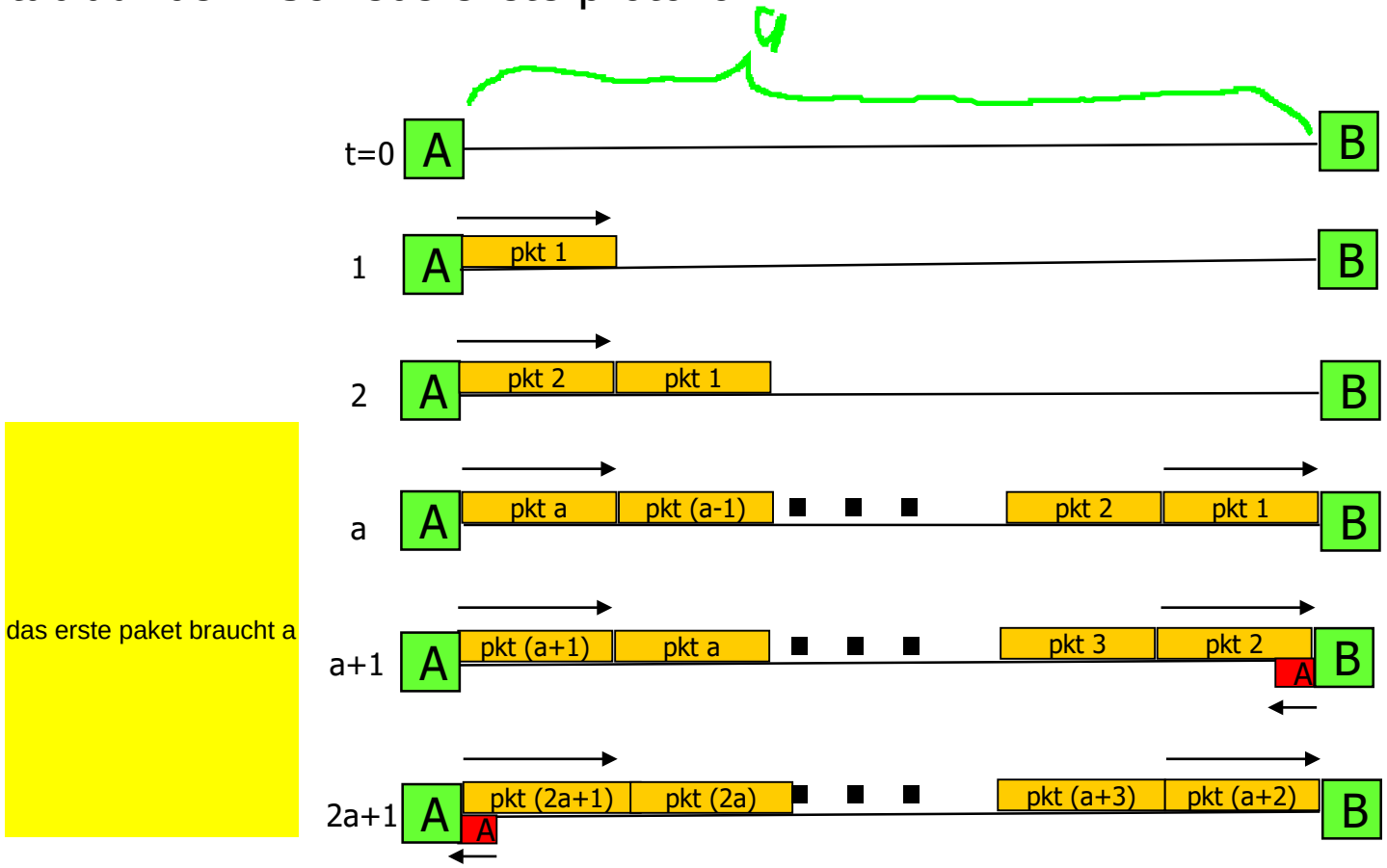
  - Fall 2: das Fenster ist **nicht** groß genug, um zu senden, bis ACK zurückkommt:
    - $W < 1 + 2a$
    - $S = \frac{W \cdot L}{L/R + 2D} \cdot \frac{1}{R} = \frac{W}{1 + 2a}$

a ist die Kanalpuffergröße

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

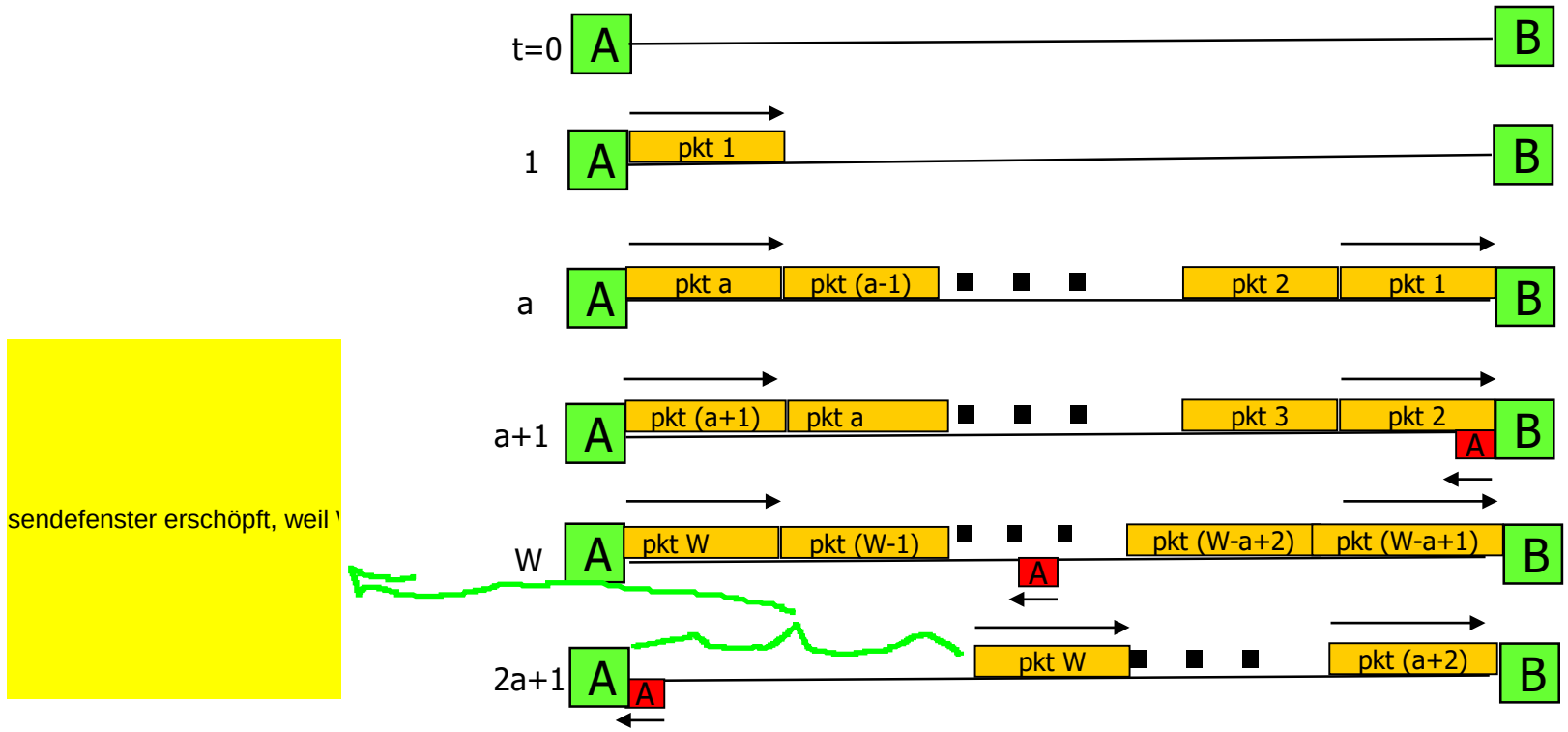
Leistungsanalyse: Schiebefensterprotokolle

- Zeitablauf beim Schiebefensterprotokoll



$W > 2a + 1$

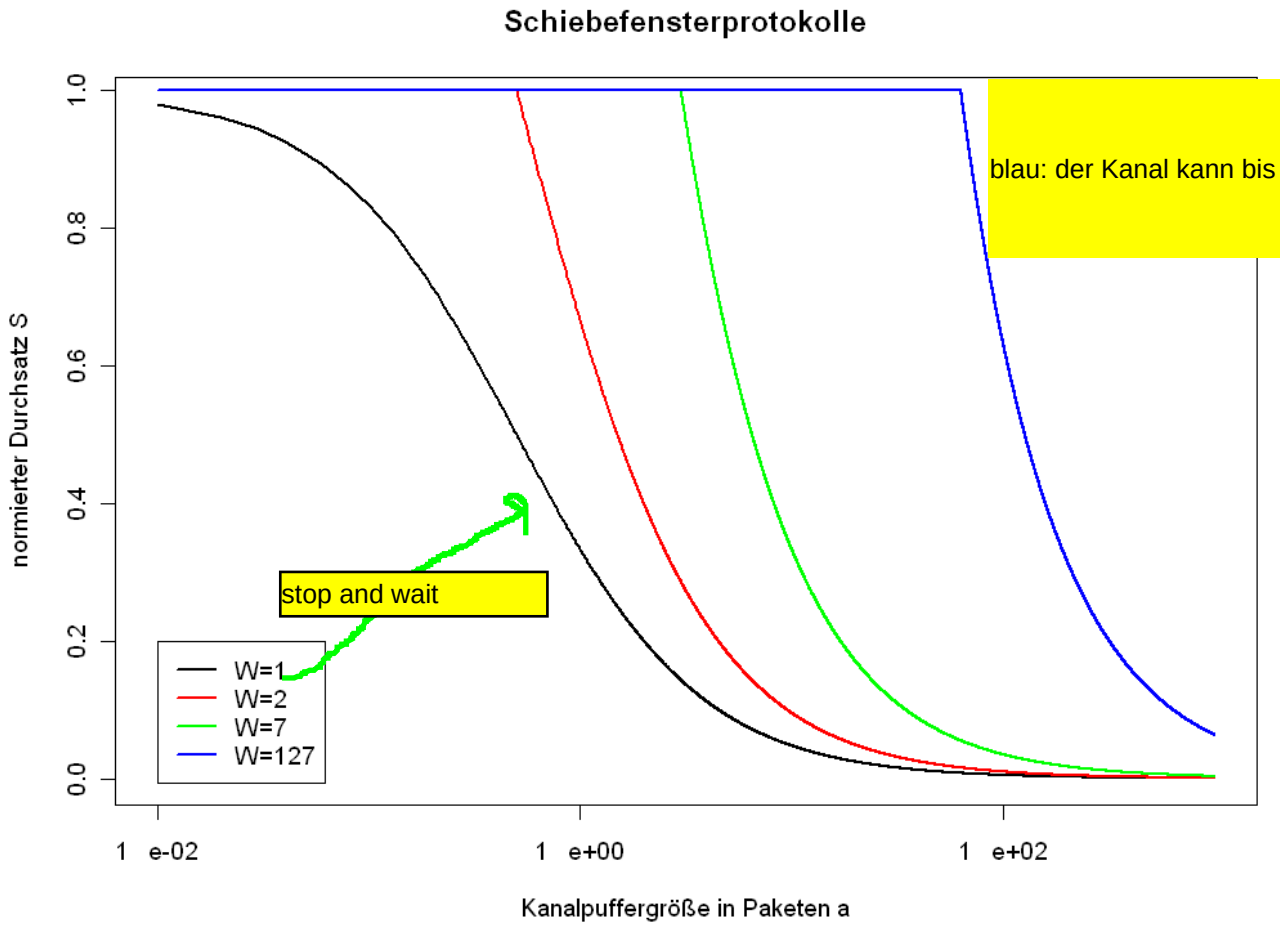
Leistungsanalyse: Schiebefensterprotokolle



$W < 2a + 1$

Leistungsanalyse: Schiebefensterprotokolle

- normierter Durchsatz von Schiebefensterprotokollen als Funktion von a:



→ eine Fenstergröße W reicht bis zu einer maximalen Größe a = (W-1)/2, um den Kanal zu füllen, danach sinkt der Durchsatz

Leistungsanalyse: Schiebefensterprotokolle

Selective Repeat mit Fehlern

- Annahme: unabhängige Fehler mit Wahrscheinlichkeit p
- N = E[Sendeversuche] = 1/(1-p) in stop-and-wait hergeleitet
- der Durchsatz im fehlerfreien Fall muss durch N geteilt werden:

$$S = \begin{cases} \frac{1}{N} = \frac{1}{1/(1-p)} = 1-p & W \geq 1+2a \\ \frac{W}{N \cdot (1+2a)} = \frac{W}{1/(1-p) \cdot (1+2a)} = \frac{W(1-p)}{1+2a} & W < 1+2a \end{cases}$$

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

also genau so wie bei STOP-and-WAIT (nur jetzt komp

Leistungsanalyse: Schiebefensterprotokolle

- Go-back-N mit Fehlern
  - jeder Fehler erfordert eine Sendewiederholung von K Paketen
  - Annahme: im Fehlerfall ist das Fenster gefüllt und alle Pakete des Fensters müssen erneut gesendet werden, dann:

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

  - wenn das fehlerhafte Paket i-mal gesendet wird, müssen insgesamt 1+(i-1)K = (1-K)+Ki Pakete gesendet werden

i-1 fehlerhafte versendungen 1 richtige

aufteilen in 2 summen

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

indexverschiebung

abl. Trick

$$= (1-K)(1-p)\sum_{i=0}^{\infty}p^{i} + K(1-p)\left(\sum_{i=0}^{\infty}p^{i}\right)' = (1-K)(1-p)\frac{1}{1-p} + K(1-p)\frac{1}{(1-p)^2}$$

geometr. Reihe

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

Leistungsanalyse: Schiebefensterprotokolle

- mit K erhalten wir:

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

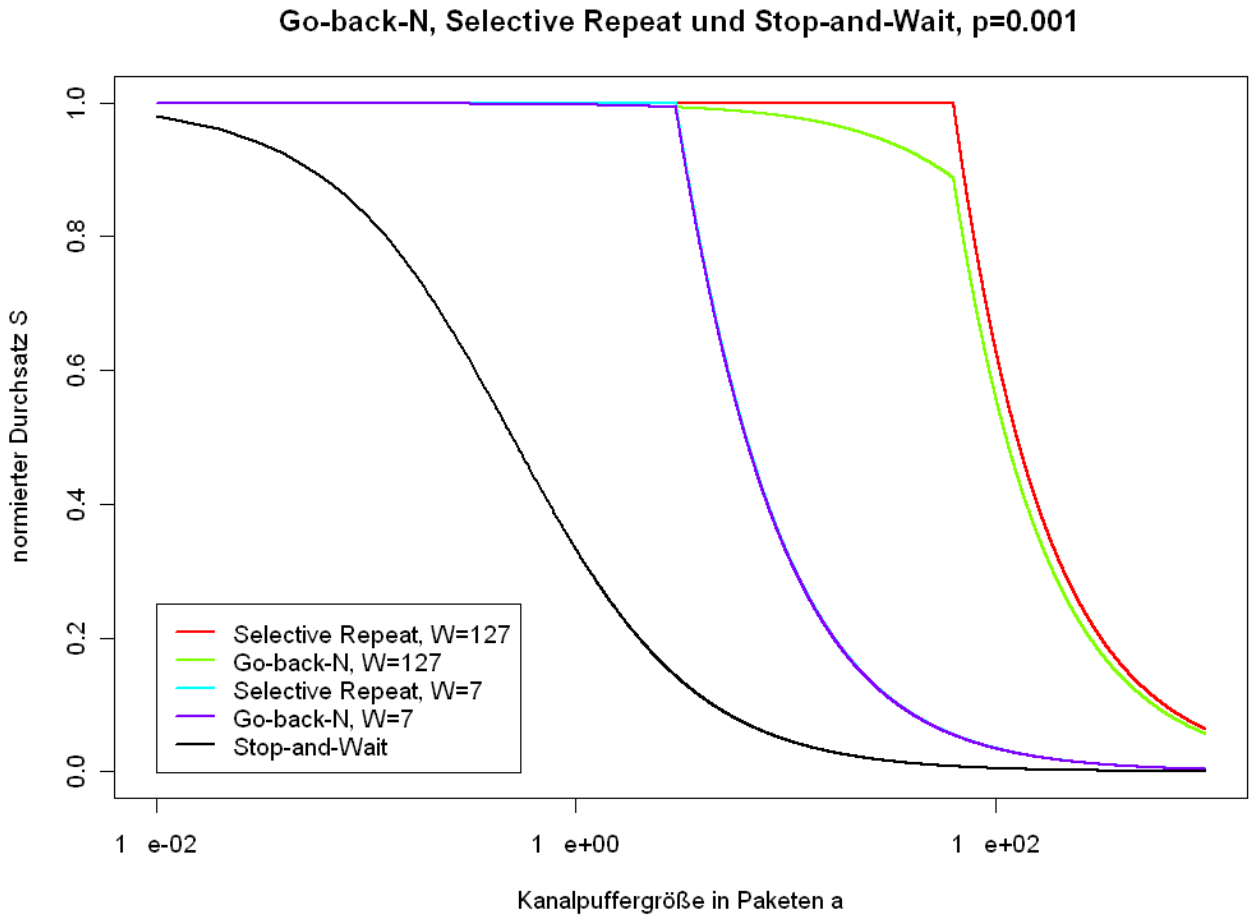
- Division des Durchsatzes ohne Fehler durch N ergibt:

$$S = \begin{cases} \frac{1}{N} = \frac{1-p}{1+2ap} & W \geq 1+2a \\ \frac{W}{N \cdot (1+2a)} = \frac{W(1-p)}{(1-p+Wp) \cdot (1+2a)} & W < 1+2a \end{cases}$$

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

Leistungsanalyse: Schiebefensterprotokolle

- normierter Durchsatz von G-Back-N und Selective Repeat als Funktion von a, $p = 10^{-3}$:

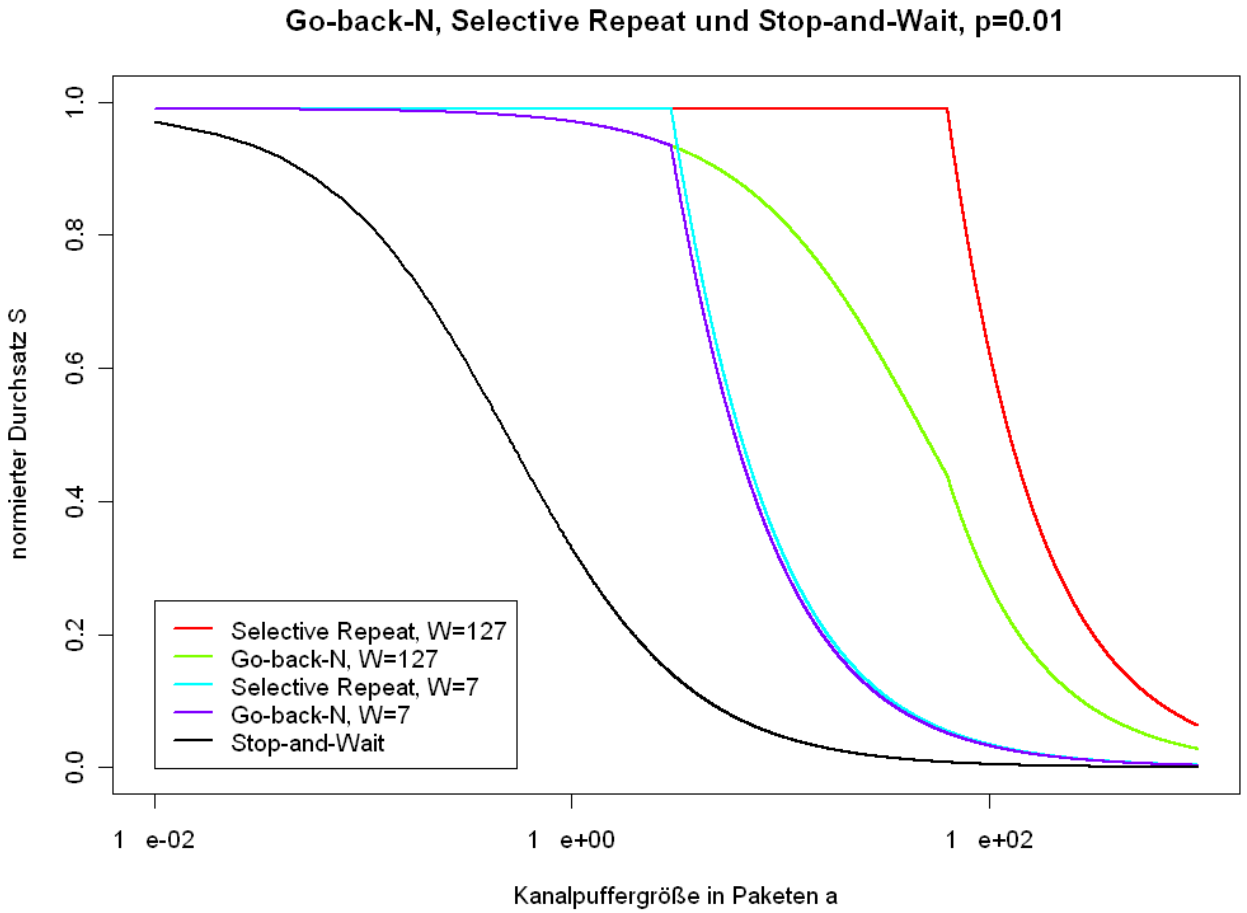


→ erst für größere Fenster ergibt sich ein spürbarer Vorteil von Selective Repeat

logisch, also bei großen fenstern sel

Leistungsanalyse: Schiebefensterprotokolle

- normierter Durchsatz von G-Back-N und Selective Repeat als Funktion von a, $p = 10^{-2}$:

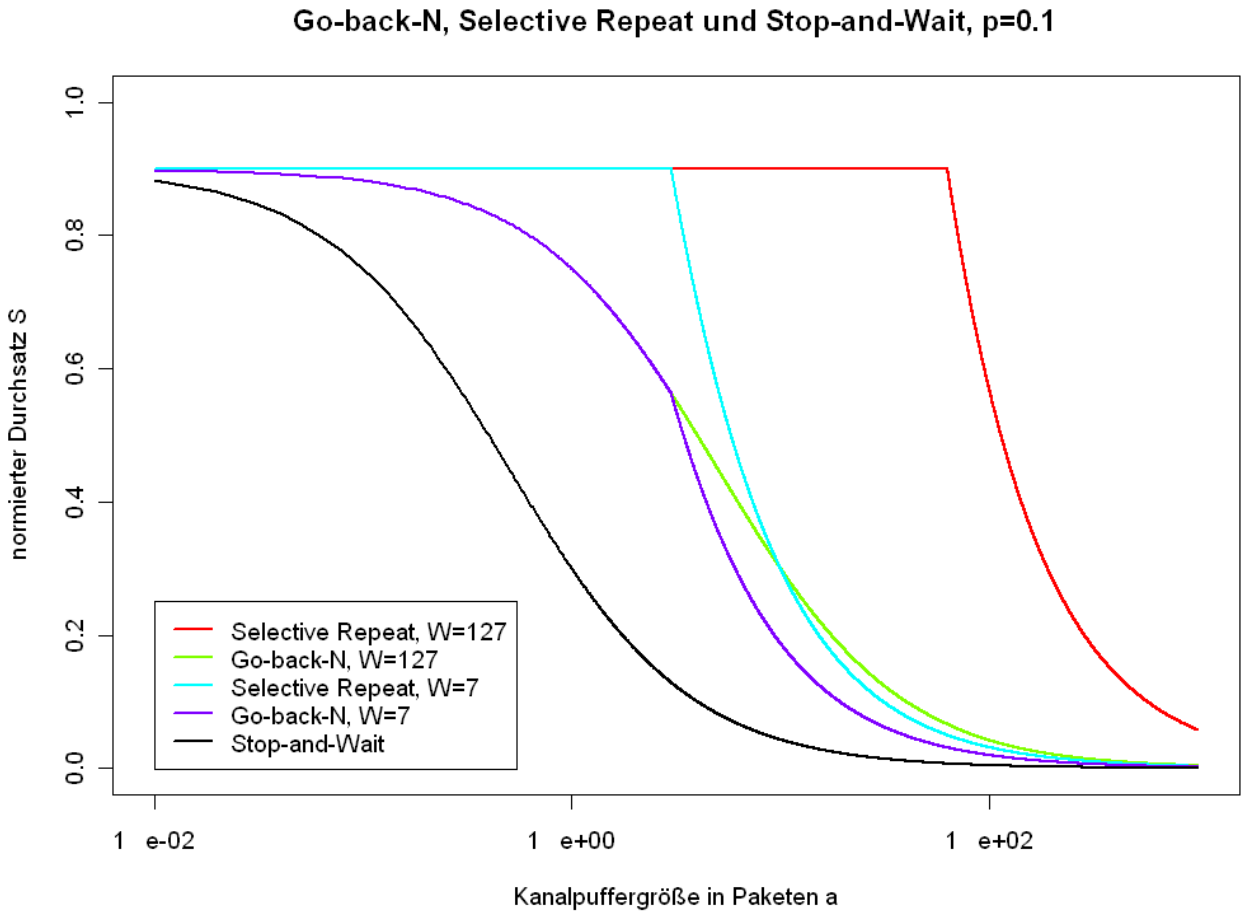


→ durch die höheren Verluste wird der der Vorteil von Selective Repeat jetzt auch bei kleineren Fenstergrößen sichtbar

bei fehlerintensiven kanälen selective repeate verwenden

Leistungsanalyse: Schiebefensterprotokolle

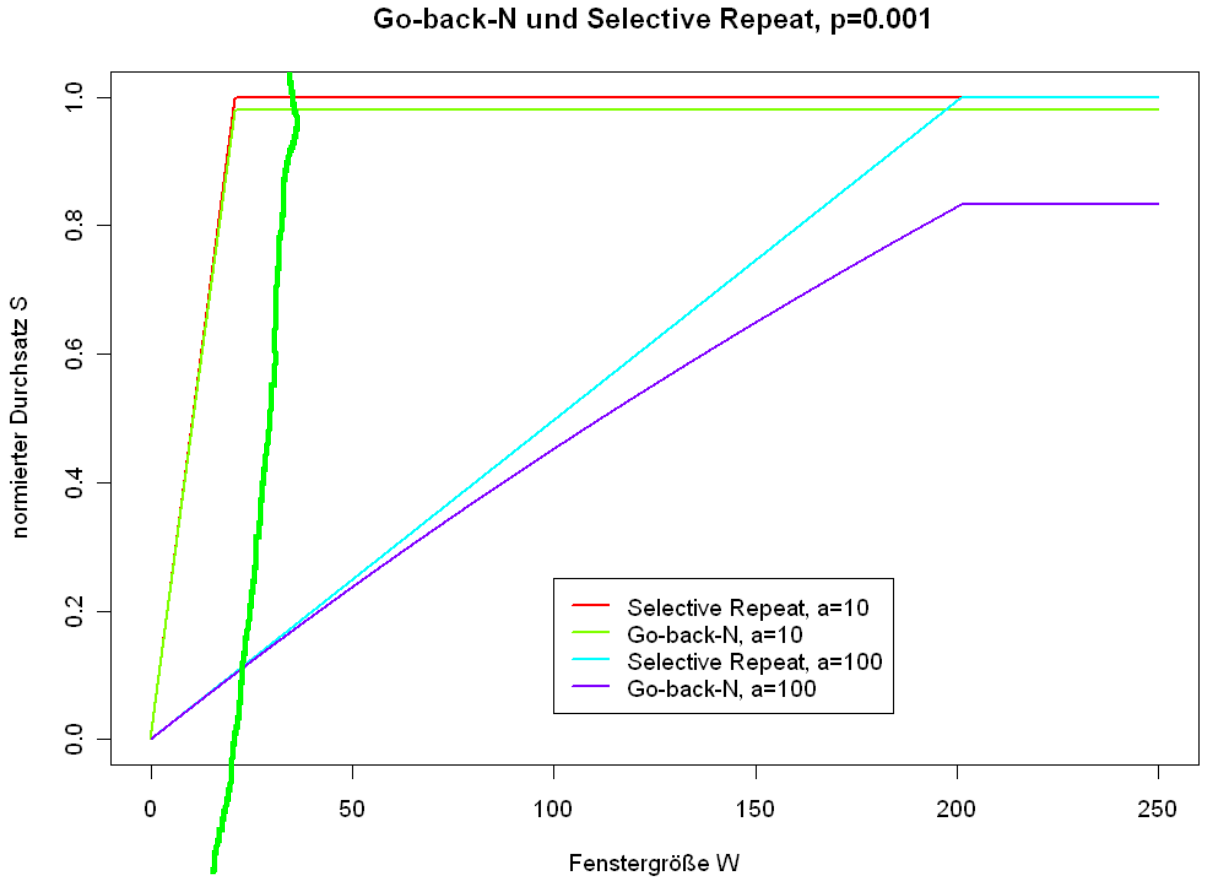
- normierter Durchsatz von G-Back-N und Selective Repeat als Funktion von a, p = 10^{-1}:



→ der Vorteil von Selective Repeat ist bei so vielen Verlusten deutlich

Leistungsanalyse: Schiebefensterprotokolle

- normierter Durchsatz von G-Back-N und Selective Repeat als Funktion von W, p = 10^{-3}:

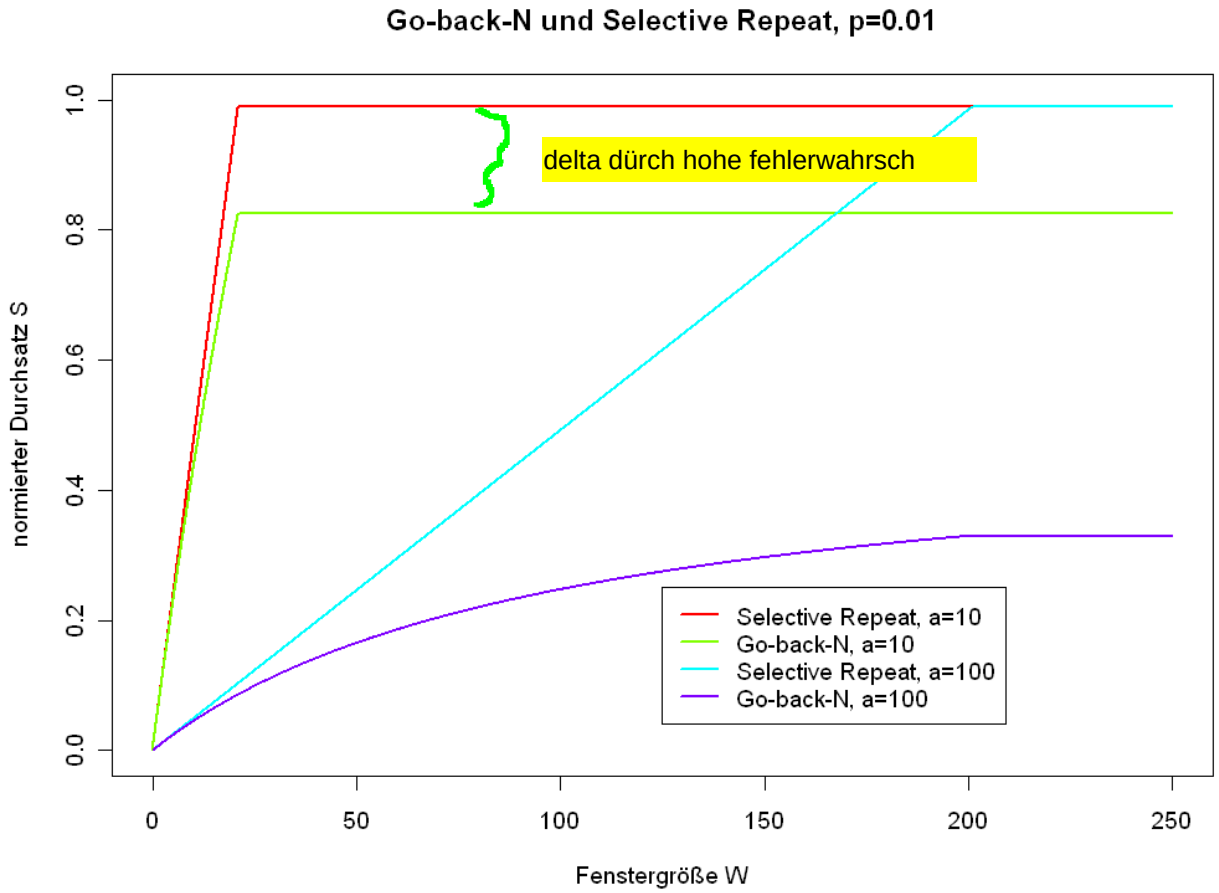


W=21 ist ein cap-off point für kanalauslastung bei 0% fehlerwarsch, weil dann der I

→ für große a und große Fenster gibt es einen erheblichen Unterschied zwischen Go-Back-N und Selective Repeat

Leistungsanalyse: Schiebefensterprotokolle

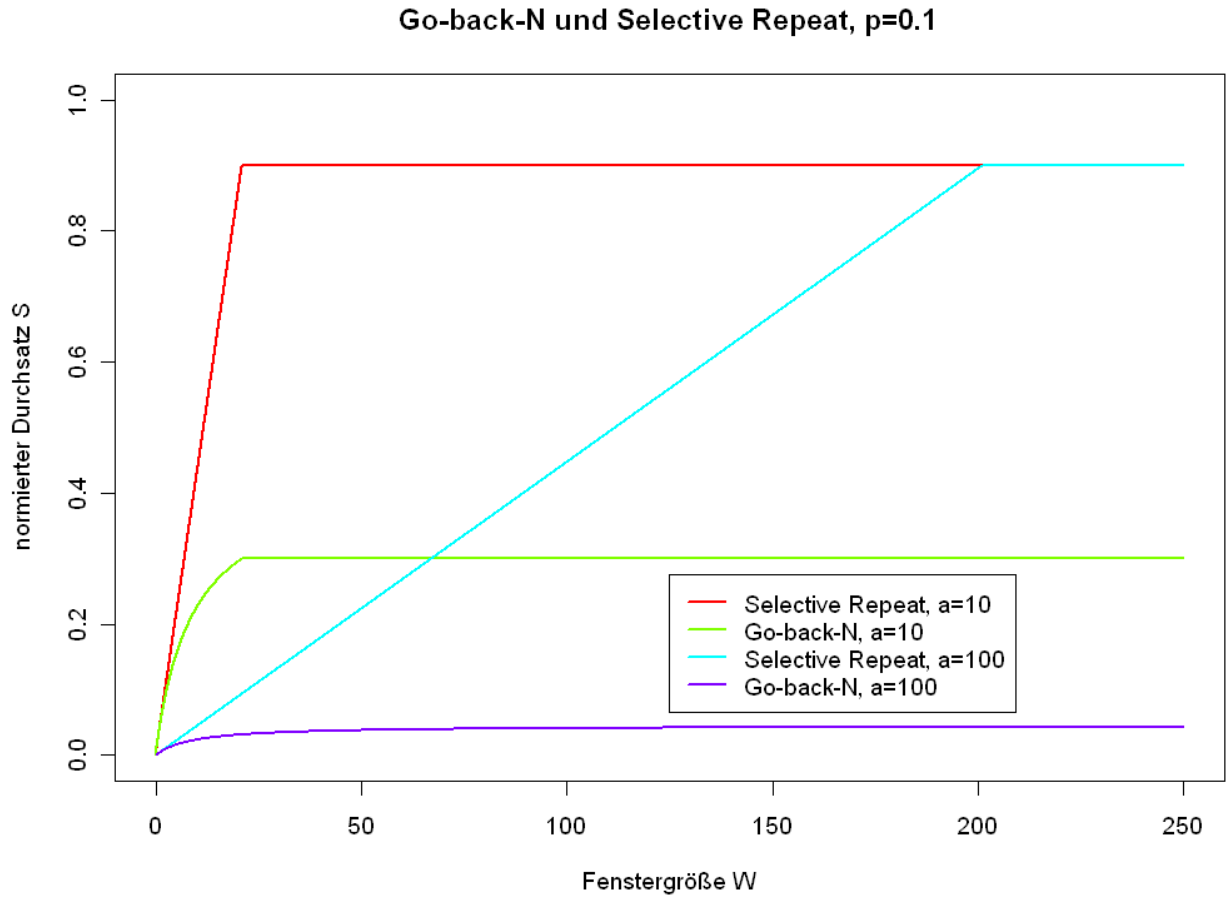
- normierter Durchsatz von G-Back-N und Selective Repeat als Funktion von W, p = 10^{-2}:



→ der Unterschied wird deutlicher

Leistungsanalyse: Schiebefensterprotokolle

- normierter Durchsatz von G-Back-N und Selective Repeat als Funktion von W, p = 10^{-1}:



→ und noch deutlicher