Vorlesungsskript Kanalcodierung II

von
DR.-ING. VOLKER KÜHN

aktualisiert von
DR.-ING. DIRK WÜBBEN

Fachbereich Physik/Elektrotechnik (FB 1)
Arbeitsbereich Nachrichtentechnik
Postfach 33 04 40
D-28334 Bremen

Version 2.4

(04.04.2011)

Kapitel 3

Verfahren zur adaptiven Fehlerkontrolle

3.1 Einführung

Bisher: FEC-Codierung (*Forward Error Correction*)

- Konstruktion möglichst leistungsfähiger Codes zur Fehlerkorrektur
- Durch feste Coderate R_c ist auch die Übertragungsrate konstant

$\longrightarrow$ **Durchsatz ist unabhängig vom Übertragungskanal!**

- Es ist kein Rückkanal erforderlich
- Nachteile:
  - Bei 'guten' Übertragungsbedingungen wird zuviel Redundanz hinzugefügt
    $\longrightarrow$ geringe Bandbreiteneffizienz
  - Bei 'schlechten' Übertragungsbedingungen reicht Korrekturfähigkeit des FEC-Codes nicht aus
    $\longrightarrow$ es treten nicht-korrigierbare Übertragungsfehler auf

$\longrightarrow$ **Qualität der Übertragung ist abhängig vom Übertragungskanal!**

Jetzt: ARQ-Verfahren (*Automatic Repeat Request*)
Unter ARQ-Verfahren versteht man Übertragungsprotokolle, die im Fall einer fehlerhaften Übertragung die falsch empfangenen Bereiche einer Nachricht wiederholen, also sozusagen Redundanz zufügen (Wiederholungscode). Da diese Redundanz aber ausschließlich im Fehlerfall eingefügt wird, spricht man von adaptiven Verfahren. Ist der Kanal sehr schlecht, treten häufig Fehler auf und es sind viele Wiederholungen erforderlich (hohe Redundanz). Bei guten Kanälen reichen dagegen sehr wenige Wiederholungen und damit eine geringe Redundanz aus. Die Redundanz paßt sich also den aktuellen Kanalbedingungen an (adaptiv)!

ARQ-Verfahren kommen in der Praxis sehr häufig zum Einsatz, und zwar immer dann, wenn sehr hohe Anforderungen an die Übertragungssicherheit, d.h. an die Fehlerrate gestellt werden. Im Zweifelsfall kann so lange wiederholt werden, bis endlich eine fehlerfreie Übertragung zustande gekommen ist. Hierdurch wird deutlich, dass die Nettodatenrate während schlechter Kanalbedingungen drastisch reduziert werden kann (s. auch Abschnitt 3.5). Folgende Bedingungen müssen für ein ARQ-System erfüllt sein:

- Es wird eine paketorientierte Übertragung (Burst-Betrieb) vorausgesetzt
- Es ist ein Rückkanal erforderlich, über den der Empfänger dem Sender mitteilen kann, dass ein Paket fehlerhaft war

- Es sind fehlererkennende Codes einzusetzen.

Bild 3.1 zeigt die allgemeine Struktur eines ARQ-Systems. Die im Empfänger ankommenden Datenpakete werden zunächst decodiert und durch die Überprüfung des Syndroms auf Fehlerfreiheit getestet. Liegt ein Übertragungsfehler vor, so fordert die ARQ-Steuerung des Empfängers die Wiederholung des korrupten Blocks an, indem ein NAK (*Negative Acknowledgement*) zum Sender übertragen wird. Dessen ARQ-Steuereinheit wertet das ARQ-Signal aus und initiiert die erneute Übertragung des entsprechenden Blocks. Im fehlerfreien Fall sendet der Empfänger ein ACK-Signal (*Acknowledgement*), wodurch der Sender erfährt, dass ein Paket fehlerfrei übertragen wurde und nicht mehr länger im Speicher gepuffert werden muß.

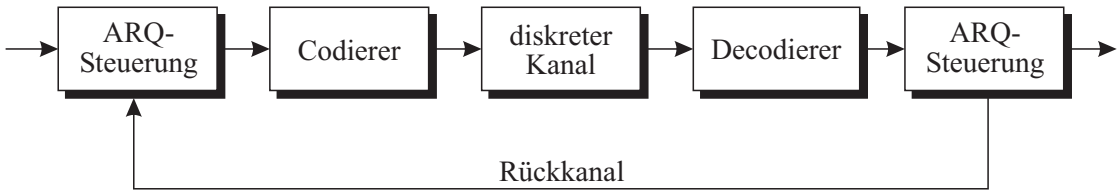

Bild 3.1: Prinzipielle Struktur eines ARQ-Systems

Fehlererkennende Codes
Wie aus dem letzten Semester bekannt ist, eignen sich CRC-Codes (*Cyclic Redundancy Check*) hervorragend zur Detektion von Bündelfehlern. Es handelt sich bei ihnen um lineare, zyklische und systematische Blockcodes, deren Generatorpolynom die Form

$$g(D) = (1 + D) \cdot p(D)$$

hat, wobei $p(D)$ ein primitives Polynom vom Grad r ist. Die Decodierung erfolgt über die Berechnung des Syndrompolynoms $s(D)$. Für $s(D) \neq 0$ wurde ein Fehler erkannt, für $s(D) = 0$ liegt entweder ein nicht erkennbarer Fehler oder aber kein Fehler vor. CRC-Codes besitzen folgende Eigenschaften:

- Alle Fehlermuster mit $w_H(\mathbf{e}) \leq 3$ werden erkannt.
- Alle Fehler mit ungeradem Gewicht werden erkannt.
- Alle Bündelfehler bis zur Länge $r + 1$ werden erkannt.
- Von den Bündelfehlern mit einer Länge von $r + 2$ wird nur eine Quote von 2^{-r} nicht erkannt
- Von den Bündelfehlern mit einer Länge von $\geq r + 3$ wird nur eine Quote von $2^{-(r+1)}$ nicht erkannt.

Es ist leicht einzusehen, dass die Leistungsfähigkeit der CRC-Codes entscheidenden Einfluß auf die Leistungsfähigkeit des gesamten ARQ-Systems hat. Werden Fehler durch ihn nicht erkannt, können die fehlerhaften Pakete auch nicht neu angefordert werden und das ARQ-System versagt.

3.2 Zuverlässigkeit der ARQ-Verfahren bei idealem Rückkanal

Es existieren zwei wichtige Qualitätsmerkmale, die ein ARQ-Prinzip charakterisieren. Zum einen ist die Zuverlässigkeit der Verfahren entscheidend, die durch die Auftrittswahrscheinlichkeit P_{ue} unerkannter Fehler bestimmt wird.

Des weiteren bestimmt der Datendurchsatz

$$\eta = \frac{\text{Anzahl fehlerfrei empfangener Infobit}}{\text{Gesamtzahl übertragener Bit}} = \frac{\text{Anzahl fehlerfrei empfangener Blöcke}}{\text{Gesamtzahl übertragener Blöcke}} \cdot R_c \qquad (3.1)$$

die Effizienz des Systems, wobei R_c die Coderate des eingesetzten fehlererkennenden Codes beschreibt. Der Durchsatz η entspricht der bekannten Coderate bei FEC-Systemen, ist hier allerdings variabel und paßt sich dem Übertragungskanal an. Er hängt von den jeweiligen ARQ-Strategien ab und wird daher in den nächsten Abschnitten für jedes Verfahren getrennt ermittelt.

In diesem Abschnitt beschäftigen wir uns mit der Zuverlässigkeit der ARQ-Systeme und gehen zunächst von einem idealen Rückkanal aus, die ACK- bzw. NAK-Signale des Empfängers erreichen also unverfälscht den Sender. Weiterhin gelten die Vereinbarungen:

P_{ue} Auftrittswahrscheinlichkeit eines unerkannten Fehlers (*undetected error*)

P_{ed} Auftrittswahrscheinlichkeit eines erkennbaren Fehlers (*error detected*)

P_w Wahrscheinlichkeit für das Versagen des ARQ-Systems (Fehler wird nicht erkannt)

Dann gilt:

$$\begin{aligned} P_w &= \underbrace{P_{ue}}_{\substack{\text{Fehler nicht}\\ \text{erkannt}}} + \underbrace{P_{ed} \cdot P_{ue}}_{\substack{\text{1 Wiederholung,}\\ \text{Fehler nicht erkannt}}} + \underbrace{P_{ed}^2 \cdot P_{ue}}_{\substack{\text{2 Wiederholungen,}\\ \text{Fehler nicht erkannt}}} + \cdots \\ &= P_{ue} \cdot \sum_{i=0}^{\infty} P_{ed}^i \\ &= \frac{P_{ue}}{1 - P_{ed}} \end{aligned} \tag{3.2}$$

Gl. (3.2) erlaubt folgende Interpretation: Die Zuverlässigkeit eines ARQ-Systems ist unabhängig von dem verwendeten Verfahren. Lediglich der zum Einsatz kommende fehlererkennende Code, z.B. CRC-Codes, bestimmt die Wahrscheinlichkeit für das Auftreten nicht erkennbarer Fehlermuster. Für einen Genie-Code, der alle Fehlermuster erkennen würde, wäre $P_{ue} = 0$ und somit auch $P_w = 0$. Der zur Fehlererkennung eingesetzte Code sollte also möglichst gut sein, damit sehr viele Fehlermuster erkannt werden. Das ARQ-Verfahren selbst hat allerdings keinen Einfluß auf die Zuverlässigkeit.

Bei FEC-Verfahren ist der Datendurchsatz konstant, die Zuverlässigkeit hängt hingegen vom Kanalzustand ab.
ARQ-Verfahren garantieren unabhängig vom Kanal eine gleich bleibende Übertragungssicherheit, ihr Durchsatz wird allerdings stark vom Kanal beeinflußt.

3.3 Klassische ARQ-Verfahren

Wir unterscheiden drei klassische ARQ-Verfahren, die im folgenden kurz erläutert werden. Sie können auch kombiniert werden, und zwar sowohl untereinander als auch mit FEC-Verfahren wie z.B. den Faltungscodes. Die letztgenannte Kombination wird im folgenden noch im Abschnitt 3.5 behandelt.

Neben der Funktionsbeschreibung der einzelnen ARQ-Strategien wird auch das zweite Leistungsmerkmal, der Datendurchsatz analysiert. Unter der Annahme eines idealen (störungsfreien) Rückkanals können wir die Effizienz η des SW-Verfahrens leicht berechnen. Wir setzen ferner voraus, dass ein Genie-Code, der alle Fehlermuster erkennt ($P_{ue} = 0 \;\rightarrow\; P_w = 0$), zum Einsatz kommt. Diese Annahme ist erlaubt, denn zur Berechnung des Datendurchsatzes spielen nicht erkannte Fehler keine Rolle, da sie nicht zu Wiederholungen führen.

3.3.1 *Stop & Wait*-Verfahren (SW)

Die einfachste, aber auch schlechteste Methode zur adaptiven Fehlerkontrolle ist das *Stop & Wait*-Verfahren. Die Funktionsweise ist in Bild 3.2 dargestellt. Rot eingefärbte Pakete sind fehlerhaft empfangen worden, blaue Pakete zeigen Wiederholungen an. Diese Darstellung wird auch in den folgenden Bildern beibehalten.

Es wird ein Block der Dauer T_B gesendet und solange mit dem Senden des zweiten Blocks gewartet, bis vom Empfänger die Bestätigung für den korrekten Empfang kommt (*ACK - Acknowledgement*). Tritt während der Übertragung ein Fehler auf, welcher im Empfänger erkannt wird, so sendet dieser ein NAK (*Negative Acknowledgement*), das den Sender zum Wiederholen des fehlerhaften Blocks veranlaßt.

Die Zeit, die zwischen zwei gesendeten Blöcken verstreicht, wird mit T_G bezeichnet und hängt von der Verzögerung der gesamten Datenübertragungsstrecke (Hin- und Rückkanal) ab (*round trip time*). Die Gesamtdauer zum Übertragen eines Blocks beträgt beim SW-Verfahren somit

$$T_t = T_B + T_G \tag{3.3}$$

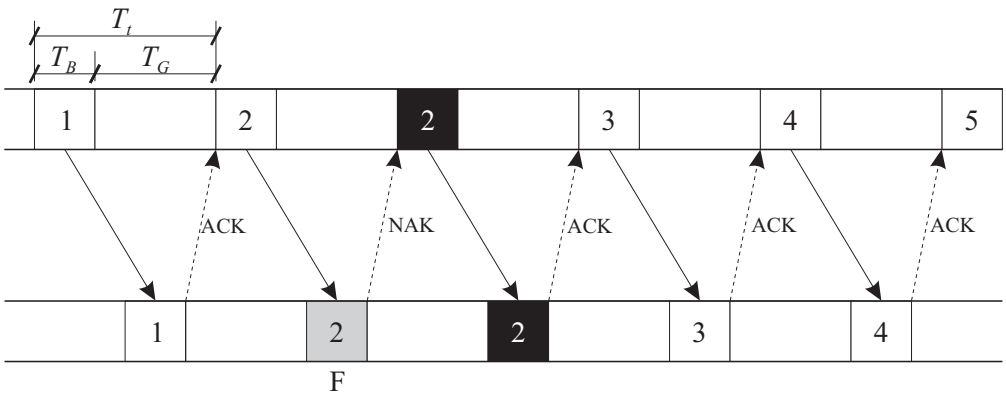

Bild 3.2: Funktionsweise des *Stop & Wait*-Verfahrens

Vorteile:

- Leicht zu implementieren
- Nur kleiner Puffer im Sender (1 Block), gar kein Puffer im Empfänger erforderlich!

Nachteile:

- Geringer Datendurchsatz durch hohe Leerlaufzeiten

Datendurchsatz bei idealem Rückkanal
Entsprechend Bild 3.2 benötigt ein fehlerfrei empfangenes Paket die Zeit $T_t = T_B + T_G$. Die Wahrscheinlich beträgt hierfür $P_c = 1 - P_{ed}$, da $P_{ue} = 0$ gilt. Werden fehlerhafte Blöcke empfangen, gilt folgender Zusammenhang:

Übertragung	$P_c = (1 - P_{ed})$	T_t
1 Wiederholung	$P_c = P_{ed}(1 - P_{ed})$	$2T_t$
2 Wiederholungen	$P_c = P_{ed}^2(1 - P_{ed})$	$3T_t$
3 Wiederholungen	$P_c = P_{ed}^3(1 - P_{ed})$	$4T_t$

Hieraus folgt für die mittlere Übertragungszeit eines Blocks:

$$\begin{aligned} T_{AV} &= (1-P_{ed})\cdot T_t + P_{ed}(1-P_{ed})\cdot 2T_t + P_{ed}^2(1-P_{ed})\cdot 3T_t + \cdots \\ &= T_t(1-P_{ed})\cdot\sum_{i=0}^{\infty}(i+1)\cdot P_{ed}^i \\ &= \frac{T_t(1-P_{ed})}{(1-P_{ed})^2} = \frac{T_t}{1-P_{ed}}\,. \end{aligned} \tag{3.4}$$

Die Effizienz η berechnet sich nun aus dem Verhältnis der Dauer eines Blocks T_B zur mittleren Übertragungszeit T_{AV} multipliziert mit der Coderate R_c des CRC-Codes. Wir erhalten

$$\begin{aligned} \eta_{SW} &= \frac{T_B}{T_{AV}}\cdot R_c = \frac{T_B}{T_t}\cdot(1-P_{ed})\cdot R_c = \frac{T_B}{T_B+T_G}\cdot(1-P_{ed})\cdot R_c \\ &= \frac{1-P_{ed}}{1+T_G/T_B}\cdot R_c\,. \end{aligned} \tag{3.5}$$

Die Durchsatzrate η wird also einerseits über P_{ed} durch den Kanal beeinflußt, andererseits aber auch durch die Systemrandbedingungen, die sich im Verhältnis T_G/T_B widerspiegeln. Geht die Fehlerwahrscheinlichkeit des Kanals und damit auch P_{ed} gegen Null, so gilt $\eta_{SW} = R_C/(1+T_G/T_B)$. Ist zusätzlich die Leerlaufzeit $T_G << T_B$, nimmt Gl. (3.5) die Form $\eta_{SW} = R_c$ an.

3.3.2 *Go-Back*-N-Verfahren (GB-N)

Das *Go-Back*-N-Verfahren stellt eine wesentliche Verbesserung gegenüber dem SW-Prinzip dar. Seine Funktionsweise zeigt Bild 3.3. Es wird nach dem Senden eines Blocks zunächst nicht mehr auf eine Antwort des Empfängers gewartet, sondern die nachfolgenden Blöcke werden direkt nacheinander übertragen. Wird im Empfänger ein Fehler detektiert und das NAK an den Sender geleitet, geht dieser bis zum fehlerhaften Block zurück (*go-back*-N) und wiederholt den falschen und auch alle nach ihm gesendeten Pakete, unabhängig davon, ob diese falsch waren oder nicht. Dies erklärt auch den Namen des Verfahrens.

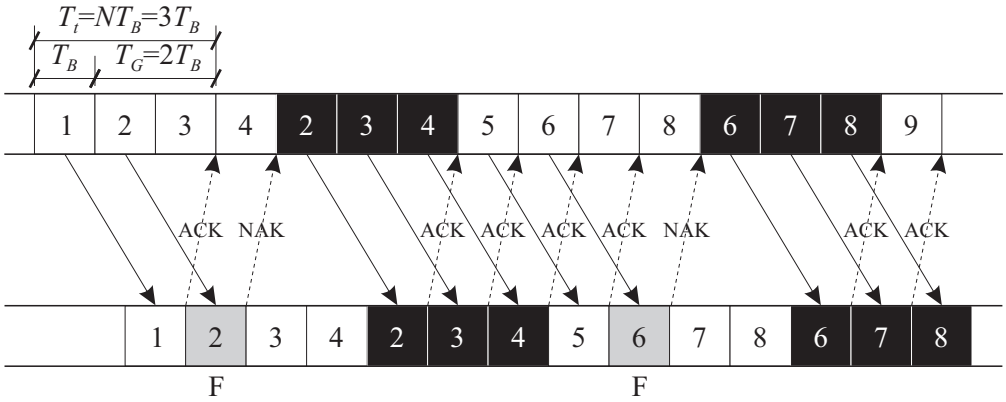

Bild 3.3: Funktionsweise des *Go-Back*-N-Verfahrens, $N = 3$

Der entscheidende Parameter $N = \lceil T_t/T_B \rceil = \lceil T_G/T_B \rceil + 1$ gibt an, wieviele Pakete im Fehlerfall zu wiederholen sind und hängt wie auch T_G vom *round trip delay* ab. Der Vorteil besteht in einem höheren Datendurchsatz gegenüber der SW-Methode. Allerdings wird im Sender mehr Speicher zum Puffern der gesendeten Blöcke benötigt, da nun die letzten N Blöcke, für die noch kein ACK empfangen wurde, gesichert werden müssen. Dies erfordert auch einen höheren Protokollaufwand.

Datendurchsatz bei idealem Rückkanal
Für das GB-N-Verfahren gilt:

Fehlerfrei	$P_c = 1 - P_{ed}$	T_B
1 Wiederholung	$P_c = P_{ed}(1 - P_{ed})$	$(N+1)T_B$
2 Wiederholungen	$P_c = P_{ed}^2(1 - P_{ed})$	$(2N+1)T_B$
3 Wiederholungen	$P_c = P_{ed}^3(1 - P_{ed})$	$(3N+1)T_B$

Wir erhalten

$$\begin{aligned} T_{AV} &= (1-P_{ed}) \cdot T_B + P_{ed}(1-P_{ed}) \cdot (N+1)T_B + P_{ed}^2(1-P_{ed}) \cdot (2N+1)T_B + \cdots \\ &= T_B(1-P_{ed}) \cdot \sum_{i=0}^{\infty}(iN+1) \cdot P_{ed}^i = T_B(1-P_{ed}) \cdot \frac{1+(N-1)P_{ed}}{(1-P_{ed})^2} \\ &= T_B \cdot \frac{1+(N-1)P_{ed}}{(1-P_{ed})} \, . \end{aligned} \tag{3.6}$$

Die Effizienz η lautet nun

$$\eta_{GB-N} = \frac{T_B}{T_{AV}} \cdot R_c = \frac{1-P_{ed}}{1+(N-1)P_{ed}} \cdot R_c \, . \tag{3.7}$$

Für $P_{ed} \to 0$ strebt η_{GB-N} gegen die Coderate R_c des CRC-Codes.

3.3.3 *Selective Repeat*-Verfahren (SR)

Das SR-Verfahren besitzt die größte Effizienz, da es kontinuierlich die Blöcke sendet und im Fehlerfall nur die wirklich fehlerhaften Pakete wiederholt. Diese Prozedur ist in Bild 3.4 illustriert.

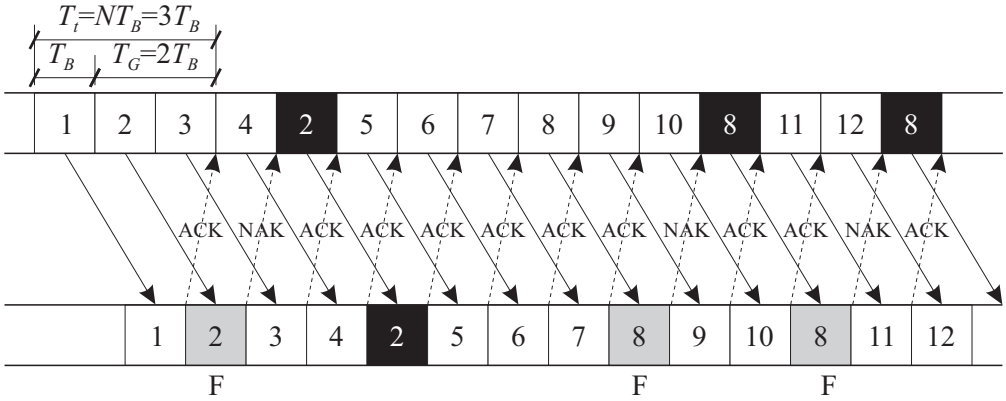

Bild 3.4: Funktionsweise des *Selective-Repeat*-Verfahrens

Es ist sofort vorstellbar, dass dieses Verfahren den größten Datendurchsatz verspricht. Allerdings ist der erhöhte Aufwand nicht zu unterschätzen. Zunächst bringt das SR-Prinzip einen Mehraufwand bzgl. der Protokollebene mit sich. Alle Pakete müssen durchnumeriert werden, damit sie im Empfänger wieder in die richtige Reihenfolge gebracht werden können und der Sender im Fehlerfall überhaupt weiß, welcher Block zu wiederholen ist.

Außerdem ist theoretisch ein unendlich großer Puffer im Empfänger erforderlich, da nach dem Empfang eines inkorrekten Blocks stets neue Pakete gesendet werden, die solange zwischengespeichert werden müssen, bis das entsprechende Paket fehlerfrei empfangen wurde. Kommt es mehrfach falsch am Empfänger an, vervielfacht sich der erforderliche Speicher. In der Realität steht im Empfänger nur ein endlich großer Speicher zur Verfügung, so dass es während schlechter Übertragungsbedingungen (viele Wiederholungen) zum Überlaufen

des Puffers und damit zu einem Datenverlust, da fehlerfrei empfangene Pakete im Sender nicht mehr gespeichert werden.

Datendurchsatz bei idealem Rückkanal
Für das SR-Verfahren gilt:

Fehlerfrei	$P_c = 1 - P_{ed}$	T_B
1 Wiederholung	$P_c = P_{ed}(1 - P_{ed})$	$2T_B$
2 Wiederholungen	$P_c = P_{ed}^2(1 - P_{ed})$	$3T_B$
3 Wiederholungen	$P_c = P_{ed}^3(1 - P_{ed})$	$4T_B$

Wir erhalten

$$\begin{aligned} T_{AV} &= (1 - P_{ed}) \cdot T_B + P_{ed}(1 - P_{ed}) \cdot 2T_B + P_{ed}^2(1 - P_{ed}) \cdot 3T_B + \cdots \\ &= T_B(1 - P_{ed}) \cdot \sum_{i=0}^{\infty}(i + 1) \cdot P_{ed}^i \\ &= T_B \cdot \frac{1 - P_{ed}}{(1 - P_{ed})^2} \\ &= \frac{T_B}{(1 - P_{ed})} \end{aligned} \tag{3.8}$$

Die Effizienz η lautet nun

$$\eta_{SR} = (1 - P_{ed}) \cdot R_c \tag{3.9}$$

Für $P_{ed} \to 0$ strebt η_{SR} gegen die Coderate R_c des CRC-Codes.

3.3.4 Kombination von *Selective Repeat*-Verfahren und *Go-Back*-N

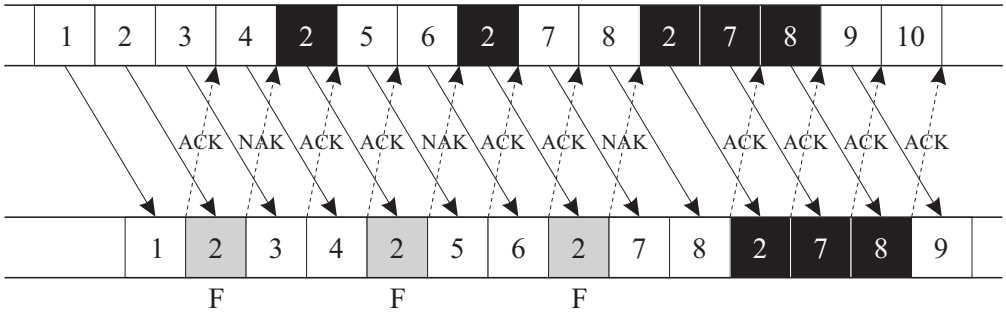

Bild 3.5: Funktionsweise des *Selective-Repeat/Go-Back-*N-Verfahrens, $N = 3$

Um diesen in der Praxis stark ins Gewicht fallenden Nachteil zu umgehen, kann eine Kombination von SR und GB-N verwendet werden. Dem liegt folgende Vorgehensweise zugrunde (s. Bild 3.5):

- Die Übertragung startet zunächst nach dem SR-Verfahren
- Wird bei schlechten Kanalbedingungen ein Block mehrmals fehlerhaft empfangen, erfolgt eine Umschaltung auf das *Go-Back-*N-Prinzip

$\longrightarrow$ Von nun an werden die N letzten Blöcke im Fehlerfall wiederholt. Dies reduziert zwar den Datendurchsatz, allerdings wird ein Pufferüberlauf im Empfänger verhindert, da zwischen den Wiederholungen keine neuen, sondern stets die gleichen Pakete übertragen werden.

3.3.5 *Selective Repeat*-Verfahren mit *Stutter*-Modus

Eine andere Möglichkeit, das Überlaufen des Puffers zu verhindern, besteht im Umschalten vom SR-Verfahren in einen *Stotter*-Modus (engl. *stutter*), der den bereits mehrfach fehlerhaft empfangenen Block solange wiederholt, bis er schließlich fehlerfrei übertragen wurde. Diese Methode ist einfacher als die letzte zu implementieren, ihr Durchsatz ist aber auch geringer, da der Stotter-Modus prinzipiell dem SW-Verfahren entspricht. Die schematische Darstellung zeigt Bild 3.6.

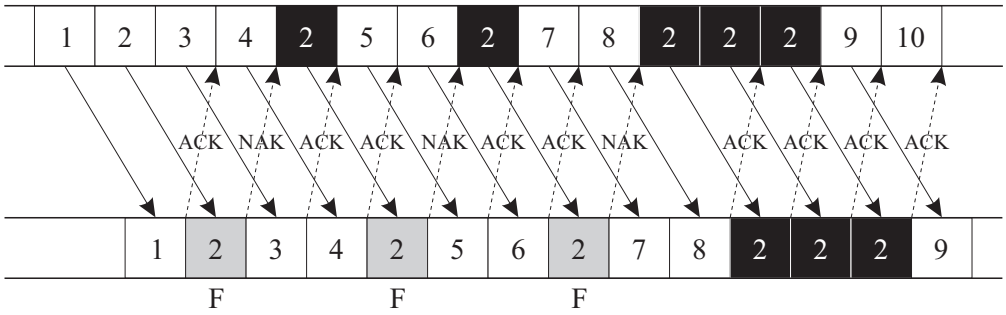

Bild 3.6: Funktionsweise des *Selective-Repeat/Stutter*-Verfahrens

3.3.6 Vergleich der ARQ-Strategien

In diesem Abschnitt sollen die Durchsatzraten η der verschiedenen ARQ-Strategien für unterschiedliche Anwendungsbereiche verglichen werden. Zum einen wurde eine typische Satellitenverbindung über geostationäre Satelliten betrachtet, zum anderen eine Richtfunkstrecke. Beide Übertragungskanäle können in guter Näherung als AWGN-Kanäle aufgefaßt werden, d.h. lediglich additives weißes gaußverteiltes Rauschen stört die Übertragung. Der wesentliche Unterschied besteht in der Entfernung zwischen Sender und Empfänger und damit im *round trip delay*.

Geostationäre Satelliten befinden sich in einer Umlaufbahn ca. 36.000 km über dem Äquator. Gehen wir von dem Szenarium aus, dass die Verbindung Erde-Satellit-Erde für Hin- und Rückkanal verwendet wird, ergibt sich eine Gesamtstrecke von

$$s = 4 \cdot 36.000 \text{ km} = 144.000 \text{ km} ,$$

die einer Laufzeit des Signals von

$$T_G = \frac{s}{c} = \frac{144 \cdot 10^6 \text{ m}}{3 \cdot 10^8 \text{m/s}} = 0.48 \text{ s} .$$

Die Effizienz der ARQ-Verfahren SW und GB-N hängt vom Verhältnis von T_G und der Paketdauer T_B ab. Nehmen wir eine Blockdauer von $T_B = 20$ ms an, wie es in der Sprachübertragung üblich ist, so gilt $N = 25$. Bei kürzeren Paketzeiten wie z.B. $T_B = 6$ ms gilt dagegen $N = 81$. Demgegenüber treten bei einer Richtfunkstrecke nahezu keine Verzögerungszeiten auf, wir nehmen daher in diesem Fall $N = 2$ an.

Bild 3.7 zeigt die Durchsatzraten η der drei ARQ-Strategien für die oben diskutierten Anwendungsbereiche. Dabei wurde ein einfacher (127,71)-BCH-Code angenommen. Der konkrete Code beeinflußt die Kurven lediglich bzgl. der Coderate und damit der asymptotischen Effizienz und hinsichtlich der Berechnung der Fehlerwahrscheinlichkeit P_{ed}.

Es ist zu erkennen, dass das *Selective Repeat*-Verfahren unabhängig von der systembedingten Verzögerungszeit ist. Für die beiden anderen Strategien gilt, dass die Effizienz η mit sinkender Verzögerungszeit T_G zunimmt. Dabei kommt das *Go Back-N*-Verfahren für $N = 2$ dem SR-Verfahren schon recht nahe. Der SW-Algorithmus besitzt aufgrund der ständig zugeführten Redundanz (T_G/T_B) mit Abstand die geringste Effizienz.

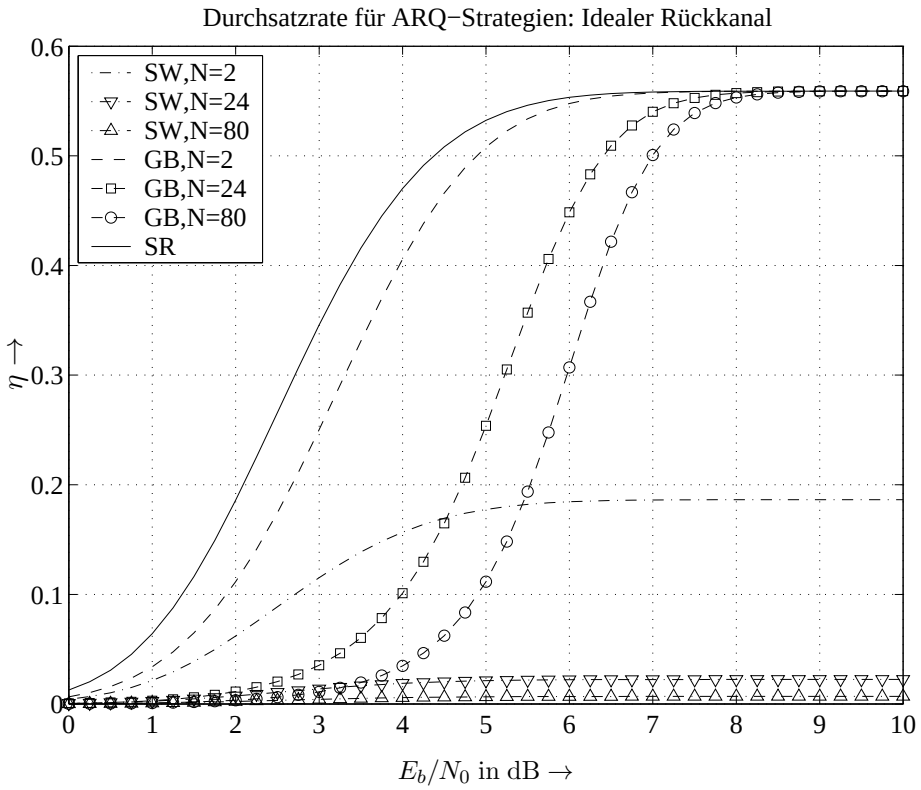

Bild 3.7: Vergleich der ARQ-Strategien

3.4 Leistungsfähigkeit bei realem Rückkanal

3.4.1 Modellbildung

Die bisherigen Ergebnisse galten alle unter der idealisierenden Annahme eines idealen (störungsfreien) Rückkanals. Dies ist in der Realität aber nie gegeben, so dass man für praktische Systeme auch Fehler im Rückkanal berücksichtigen muss. Diese Fehler führen dazu, dass vom Empfänger gesendete NAK- und ACK-Signale entweder miteinander vertauscht werden (NAK $\rightarrow$ ACK bzw. ACK $\rightarrow$ NAK) oder aber komplett verloren gehen und damit gar nicht den Sender erreichen. Daher werden i.a. folgende Gegenmaßnahmen ergriffen:

- **Zeitreferenz im Sender:**
  trifft die Antwort des Empfängers über einen gesendeten Block nicht in einem definierten Zeitintervall ein, so wird sicherheitshalber von einer erneuten Anforderung ausgegangen und der entsprechende Block erneut übertragen.
- **Zeitreferenz im Empfänger:**
  Falls nach einem gesendeten NAK der angeforderte Block nicht in einer definierten Zeit erneut am Empfänger ankommt, wird erneut ein NAK gesendet.
- Falls ein Codewort im Empfänger ankommt, dass bereits als korrekt deklariert wurde, wird abermals ein ACK gesendet und das Duplikat verworfen.

In der jetzt folgenden Analyse wollen wir annehmen, dass keine NAK-/ACK-Signale verloren gehen, sondern dass sie nur miteinander vertauscht werden können. Wir stellen uns dann das Übertragungssystem durch ein Zustandsdiagramm vor (Mealy-Automat), dessen Zustandsübergängen die Parameter a bis g zugeordnet sind (s. Bild 3.8). Welche Werte a bis g konkret annehmen, hängt von der jeweils gesuchten Zielfunktion ab.

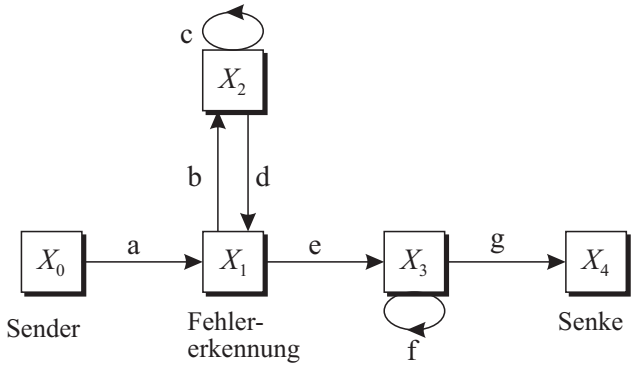

Bild 3.8: Zustandsdiagramm zur Berücksichtigung des realen Rückkanals

Der Sender (X_0) überträgt einen Datenblock (a), der im Empfänger zunächst durch Syndromdecodierung auf Fehlerfreiheit überprüft wird (X_1). Wird ein Fehler erkannt (b), muss ein NAK-Signal an den den Sender zurückgegeben werden (X_2). Wird das NAK-Signal auf dem Rückkanal verfälscht, muss es erneut übertragen werden (c), andernfalls kommt der wiederholte Block am Empfänger an (d) und die Fehlererkennung beginnt von neuem. Wird kein Fehler detektiert (e), gelangen wir in den Zustand X_3, wo nun das ACK-Signal an den Sender geschickt wird. Wird dieses verfälscht (f), erfolgt eine Wiederholung, ansonsten (g) gelangen wir zur Senke X_4.

Je nachdem, ob nun Zuverlässigkeit oder der Datendurchsatz der verschiedenen ARQ-Strategien bestimmt werden sollen, sind unterschiedliche Werte für die Parameter a bis g anzusetzen. Das Ziel besteht in allen Fällen darin, die Übertragungsfunktion vom Sender X_0 zur Senke X_4 zu bestimmen. Dazu existieren verschiedene Lösungsansätze. Eine Möglichkeit besteht in der Mason'schen Pfad-Schleifen-Regel, die ganz allgemein für beliebige Netzwerke angesetzt werden kann. Für dieses einfache Beispiel ist es aber ausreichend, das lineare Gleichungssystem aufzustellen und zu lösen.

$$\begin{aligned}
X_1 &= aX_0 + \quad + dX_2 + \quad \Rightarrow \quad X_1 = aX_0 + \tfrac{bd}{1-c}X_1 \\
X_2 &= \quad + bX_1 + cX_2 + \quad \Rightarrow \quad X_2 = \tfrac{b}{1-c}X_1 \\
X_3 &= \quad + eX_1 + \quad + fX_3 \quad \Rightarrow \quad X_3 = \tfrac{e}{1-f}X_1 \\
X_4 &= \quad + \quad + \quad + gX_3 \quad \Rightarrow \quad X_4 = \tfrac{eg}{1-f}X_1
\end{aligned}$$

Wir erhalten schließlich folgende Übertragungsfunktion

$$H = \frac{X_4}{X_0} = \frac{aeg(1-c)}{(1-f)(1-c-bd)} \; . \tag{3.10}$$

3.4.2 Zuverlässigkeit bei realem Rückkanal

In Abschnitt 3.2 wurde gezeigt, dass die Zuverlässigkeit eines ARQ-Systems unabhängig von den Kanaleigenschaften ist und lediglich von der Leistungsfähigkeit des eingesetzten fehlererkennenden Codes beeinflußt wird. Es gilt nun zu überprüfen, ob dies auch für reale, d.h. gestörte Rückkanäle gilt. Dazu setzten wir folgende Ausdrücke für die Parameter ein:

$$a = 1 \tag{3.11}$$

$$b = P_{ed} \tag{3.12}$$

$$c = P_{NA} \qquad \text{Wahrscheinlichkeit für Fehler NAK } \rightarrow \text{ ACK} \tag{3.13}$$

$$d = 1 - P_{NA} \quad \text{Wahrscheinlichkeit für korrekten Empfang von NAK} \tag{3.14}$$

$$e = P_{ue} \tag{3.15}$$

$$f = P_{AN} \quad \text{Wahrscheinlichkeit für Fehler ACK } \rightarrow \text{ NAK} \tag{3.16}$$

$$g = 1 - P_{AN} \quad \text{Wahrscheinlichkeit für korrekten Empfang von ACK} \tag{3.17}$$

Ein erkennbarer Fehler tritt mit der Wahrscheinlichkeit P_{ed} auf. In diesem Fall wird ein NAK-Signal gesendet, welches mit der Wahrscheinlichkeit P_{NA} auf dem Rückkanal verfälscht wird. Dann bleibt die Wiederholung des fehlerhaften Datenpakets aus und das NAK muss erneut übertragen werden. Die Prozedur wiederholt sich solange (Selbstschleife c in X_2), bis das NAK korrekt am Sender angekommen ($d = 1 - P_{NA}$) ist und dieser das fehlerhafte Paket erneut überträgt. Dann beginnt in X_1 die Fehlererkennung von neuem.

Die Wahl von $e = P_{ue}$ in Gl. (3.15) bedeutet, dass nur die Fälle betrachtet werden, in denen der fehlererkennende Code versagt. Gl. (3.10) liefert dann die Restfehlerwahrscheinlichkeit. Je kleiner sie ist, desto größer ist die Zuverlässigkeit. Wird kein Fehler detektiert, muss das ACK-Signal gesendet werden, das aber auch verfälscht werden kann und dann ebenfalls wiederholt werden muss (Selbstschleife f in X_3). Dies hat für die Zuverlässigkeit des Systems zunächst keine Bedeutung, sehr wohl aber für den Datendurchsatz, der in den folgenden Abschnitten betrachtet wird.

Setzen wir die oben aufgeführten Parameter in Gl. (3.10) ein, ergibt sich

$$H = \frac{P_{ue}(1-P_{AN})(1-P_{NA})}{(1-P_{AN})(1-P_{NA}-P_{ed}(1-P_{NA}))} = \frac{P_{ue}(1-P_{NA})}{(1-P_{ed})(1-P_{NA})} = \frac{P_{ue}}{1-P_{ed}} \, . \tag{3.18}$$

Dies ist das schon aus Gl. (3.2) bekannte Resultat. Damit wird deutlich, dass auch der reale Rückkanal keinen Einfluß auf die Zuverlässigkeit, also die Restfehlerwahrscheinlichkeit des ARQ-Systems, hat.

3.4.3 Datendurchsatz beim SW-Verfahren

Soll der Datendurchsatz, also die Effizienz der ARQ-Systeme berechnet werden, kommt es darauf an, wie lange die Übertragung eines Datenpakets **im Mittel** dauert. Wir definieren dazu das Zeitmaß

$$\kappa = \frac{T_B + T_G}{T_B} \, , \tag{3.19}$$

das die auf die Paketdauer T_B normierte Übertragungsdauer eines Pakets $T_B + T_G$ im fehlerfreien Fall beschreibt. Damit sich durch die Multiplikation der Übergangsparameter die Zeiten der einzelnen Vorgänge addieren, wird ein Platzhalter D eingeführt, dessen Exponent den zeitlichen Einfluß eines Zustandsübergangs darstellt. Wir erhalten folgende Zuordnungen:

$$a = D^{\kappa} \tag{3.20}$$

$$b = P_{ed} \tag{3.21}$$

$$c = P_{NA} \cdot D^{\kappa} \tag{3.22}$$

$$d = (1 - P_{NA}) \cdot D^{\kappa} \tag{3.23}$$

$$e = 1 - P_{ed} \tag{3.24}$$

$$f = P_{AN} \cdot D^{\kappa} \tag{3.25}$$

$$g = 1 - P_{AN} \tag{3.26}$$

Die Übertragung eines Blocks dauert bezogen auf T_B beim SW-Verfahren κ Zeiteinheiten. Gleiches gilt für die Übertragung der NAK- und ACK-Signale. Werden sie auf dem Rückkanal verfälscht, wir ein fehlerhafter

Block nicht wiederholt (der tatsächlich gesendete wird verworfen) oder aber ein korrekter Block unnötigerweise mehrfach übertragen, was den Datendurchsatz reduziert, d.h. die effektive Dauer für die Übertragung eines Blocks erhöht.

Die Wahl von $e = 1 - P_{ed}$ bedeutet, dass wir als fehlererkennenden Code einen Genie-Code annehmen, der alle möglichen Fehler erkennt. Sie resultiert aus der Überlegung, dass bei einem nicht erkannten Fehler das System sowieso versagt und somit die aufgetretenen Verzögerungen uninteressant sind. Bzgl. der Effizienz konzentrieren wir uns somit auf den Fall, dass das System perfekt funktioniert. Wir erhalten

$$H_{SW}(D) = \frac{(1-P_{ed})(1-P_{AN})(1-P_{NA}D^{\kappa})D^{\kappa}}{(1-P_{NA}D^{\kappa}-P_{ed}(1-P_{NA})D^{\kappa})(1-P_{AN}D^{\kappa})} \tag{3.27}$$

Die mittlere Übertragungszeit je Block steckt implizit in den Exponenten von D. Wie schon bei der Transferfunktion von Faltungscodes müssen wir Gl. (3.27) nach D differenzieren und die Ableitung an der Stelle $D = 1$ auswerten. Für die mittlere Paketübertragungsdauer ergibt sich

$$\frac{T_{AV}}{T_B} = \left.\frac{\partial H_{SW}(D)}{\partial D}\right|_{D=1} = \frac{\kappa(1-P_{ed}P_{AN}-P_{NA}+P_{ed}P_{NA})}{(1-P_{ed})(1-P_{AN})(1-P_{NA})} \tag{3.28}$$

Die Durchsatzrate beinhaltet auch noch die Coderate des Kanalcodierungsverfahrens, so dass die Effizienz der *Stop&Wait*-Strategie insgesamt

$$\eta = \frac{T_B}{T_{AV}} \cdot R_c = \frac{(1-P_{ed})(1-P_{AN})(1-P_{NA})}{(1+\frac{T_G}{T_B})(1-P_{ed}P_{AN}-P_{NA}+P_{ed}P_{NA})} \cdot R_c \tag{3.29}$$

lautet. Betrachten wir nun zwei Spezialfälle, zum einen den eines idealen Rückkanals ($P_{AN} = P_{NA} = 0$), zum anderen den eines symmetrischen Rückkanals, d.h. ($P_{AN} = P_{NA}$):

$$P_{AN} = P_{NA} = 0 \quad \Longrightarrow \quad \eta = \frac{1-P_{ed}}{1+\frac{T_G}{T_B}} R_c = \eta_I$$

$$P_{AN} = P_{NA} \quad \Longrightarrow \quad \eta = \frac{(1-P_{ed})(1-P_{AN})}{1+\frac{T_G}{T_B}} R_c = (1-P_{AN})\eta_I$$

3.4.4 Datendurchsatz beim GB-N-Verfahren

Für das *Go-Back-N*-Verfahren sind die Parameter a bis g etwas anders zu wählen. Zunächst entfällt die Leerlaufzeit T_G, da die Datenpakete ohne Pause hintereinander gesendet werden. Im fehlerfreien Fall dauert die Übertragung eines Blocks also nur noch T_B (anstatt $T_B + T_G$). Tritt ein erkennbarer Fehler auf, werden dagegen N Blöcke erneut übertragen, nicht mehr nur der fehlerhafte Block (siehe d). Wird das dazu erforderliche NAK-Signal durch den Rückkanal verfälscht, muss es erneut gesendet werden. Dies führt dazu, dass ein Block mehr wiederholt werden muss, weshalb dieser Fall zu einer Verzögerung von einer Paketdauer führt (siehe c). Die Annahme eines idealen Genie-Codes wird weiterhin aufrecht gehalten.

$$a = D \tag{3.30}$$

$$b = P_{ed} \tag{3.31}$$

$$c = P_{NA} \cdot D \tag{3.32}$$

$$d = (1-P_{NA}) \cdot D^N \tag{3.33}$$

$$e = 1 - P_{ed} \tag{3.34}$$

$$f = P_{AN} \cdot D^N \tag{3.35}$$

$$g = 1 - P_{AN} \tag{3.36}$$

Wir erhalten folgende Übertragungsfunktion:

$$H_{GB-N}(D) = \frac{(1-P_{ed})(1-P_{AN})(1-P_{NA}D)D}{(1-P_{NA}D-P_{ed}(1-P_{NA})D^N)(1-P_{AN}D^N)} \tag{3.37}$$

Die mittlere Übertragungszeit pro erfolgreich übertragenem Paket lautet

$$\frac{T_{AV}}{T_B} = \frac{1-P_{NA}-P_{ed}P_{AN}+P_{ed}P_{NA}+(N-1)(P_{AN}+P_{ed}-P_{AN}P_{NA}-P_{ed}P_{NA}-2P_{ed}P_{AN}+2P_{ed}P_{AN}P_{NA})}{(1-P_{ed})(1-P_{AN})(1-P_{NA})} \tag{3.38}$$

und die spektrale Effizienz beträgt

$$\eta = \frac{(1-P_{ed})(1-P_{AN})(1-P_{NA})}{1-P_{NA}-P_{ed}P_{AN}+P_{ed}P_{NA}+(N-1)(P_{AN}+P_{ed}-P_{AN}P_{NA}-P_{ed}P_{NA}-2P_{ed}P_{AN}+2P_{ed}P_{AN}P_{NA})} R_c \; . \tag{3.39}$$

Für die schon im letzten Abschnitt betrachteten Spezialfälle $P_{AN} = P_{NA}$ und $P_{AN} = P_{NA} = 0$ erhalten wir

$$P_{AN} = P_{NA} \quad \Longrightarrow \quad \eta = \frac{(1-P_{ed})(1-P_{AN})}{1+(N-1)(P_{AN}+P_{ed}-2P_{AN}P_{ed})} R_c$$

$$P_{AN} = P_{NA} = 0 \quad \Longrightarrow \quad \eta = \frac{1-P_{ed}}{1+(N-1)P_{ed}} R_c$$

3.4.5 Datendurchsatz beim SR-Verfahren

Für das *Selective-Repeat*-Verfahren werden die Parameter wie folgt festgesetzt:

$$a = D \tag{3.40}$$

$$b = P_{ed} \tag{3.41}$$

$$c = P_{NA} \tag{3.42}$$

$$d = (1-P_{NA}) \cdot D \tag{3.43}$$

$$e = 1-P_{ed} \tag{3.44}$$

$$f = P_{AN} \cdot D \tag{3.45}$$

$$g = 1-P_{AN} \tag{3.46}$$

Die leichten Unterschiede zum GB-N-Verfahren sind so zu erklären: Wird im Fehlerfall das NAK in ein ACK verfälscht, läuft die Übertragung irrtümlicherweise ohne Wiederholung weiter. Dies führt aber nicht zu einer zusätzlichen Verzögerung, da keine bereits korrekt empfangenen Pakete verworfen werden (siehe c, lediglich Überlauf des Puffers möglich), sondern nur das betroffene Paket wiederholt wird (siehe d). Die versehentliche Wiederholung eines schon korrekt übertragenen Blocks verursacht dagegen für beide ARQ-Strategien eine Verzögerung von einer Paketdauer.

Wir erhalten folgende Übertragungsfunktion:

$$H_{SR}(D) = \frac{(1-P_{ed})(1-P_{AN})(1-P_{NA})D}{(1-P_{NA}-P_{ed}(1-P_{NA})D)(1-P_{AN}D)} \tag{3.47}$$

Die mittlere Übertragungszeit pro erfolgreich übertragenem Paket lautet

$$\frac{T_{AV}}{T_B} = \frac{1-P_{ed}P_{AN}}{(1-P_{ed})(1-P_{AN})} \tag{3.48}$$

und die spektrale Effizienz beträgt

$$\eta = \frac{(1-P_{ed})(1-P_{AN})}{1-P_{ed}P_{AN}} R_c \; . \tag{3.49}$$

3.4.6 Vergleich der ARQ-Strategien

Beim Vergleich der ARQ-Strategien soll zunächst erst einmal der Einfluß des störungsbehafteten Rückkanals auf die jeweiligen Verfahren untersucht werden. Dazu zeigen die Bilder 3.9 bis 3.11 die Datendurchsätze η für verschiedene *round trip delays* N verschiedene Fehlerwahrscheinlichkeiten P_{AN}. Es ist ersichtlich, dass das *Stop & Wait*-Verfahren selbst bei zuverlässigen Rückkanälen ($P_{AN} = 10^{-4}$) den geringsten Datendurchsatz besitzt (vgl. idealer Rückkanal). Dementsprechend wirkt sich auch eine Degradation der Qualität des Rückkanals kaum aus.

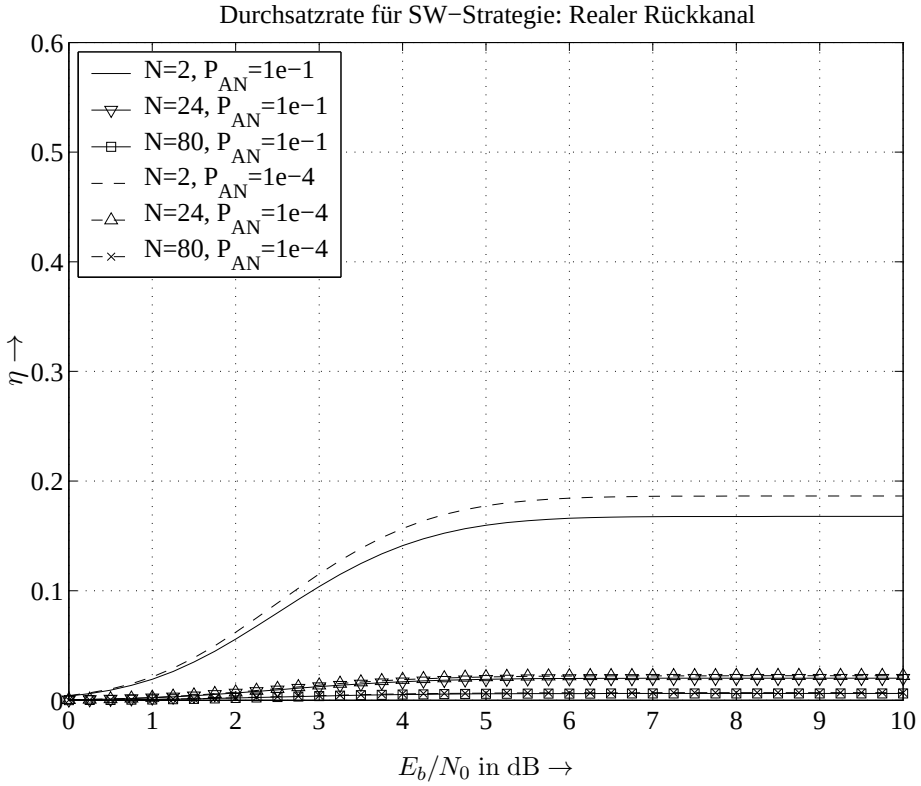

Bild 3.9: Vergleich verschiedener Fehlerwahrscheinlichkeiten P_{AN} für die SW-Strategie

Beim GB-N-Verfahren nähern sich die Kurven verschiedener N für einen zuverlässigen Rückkanal ($P_{AN} = 10^{-4}$) asymptotisch an. Weist der Rückkanal allerdings ein hohes Fehlerpotential auf ($P_{AN} = 10^{-2}$), so wird die Effizienz nachhaltig reduziert, d.h. selbst für gegen unendlich strebende Signal-Rausch-Abstände wird nicht die gleiche Effizienz erreicht wie für gute Rückkanäle. Dieser Effekt wirkt sich um so stärker aus, je größer der Parameter N, also die Anzahl der zu wiederholenden Blöcke, ist.

Das SR-Verfahren besitzt die geringste Empfindlichkeit gegenüber gestörten Rückkanälen. Dies war auch zu erwarten, da bei unnötig wiederholten Blöcken (ACK $\rightarrow$ NAK) nicht gleich N Pakete wiederholt werden, sondern nur ein einziger, wodurch sich der Durchsatz nicht dramatisch reduziert. Für den Fall (NAK $\rightarrow$ ACK) wird der empfangene Block im Empfänger nicht verworfen, sondern zwischengespeichert, so dass dies die Effizienz gar nicht beeinflußt (siehe Unabhängigkeit von η von P_{NA} in Gl. (3.49)). Allerdings bleibt festzustellen, dass die SR-Strategie nur in der Realität so nicht umzusetzen ist, da ein unendlich großer Puffer benötigt würde.

Für Übertragungssysteme mit kurzen Reichweiten, z.B. die erwähnte Richtfunkstrecke mit $N = 2$, zeigt Bild 3.12 einen direkten Vergleich der diskutierten ARQ-Strategien. Es ist erkennbar, dass der SR-Verfahren die größte Effizienz besitzt. Für geringe Qualitäten des Rückkanals nimmt der Unterschied zu. Ein Vergleich zwischen SR und GB-N zeigt, dass für $P_{AN} = 10^{-4}$ die Datendurchsätze asymptotisch gleich sind, während für $P_{AN} = 10^{-2}$ ein konstanter Unterschied von etwa 0.05 verbleibt. Für höhere Verzögerungszeiten ($N = 25$ bzw. $N = 81$) wäre der Unterschied zwischen SR auf der einen und SW bzw. GB-N auf der anderen Seite noch größer ausgefallen.

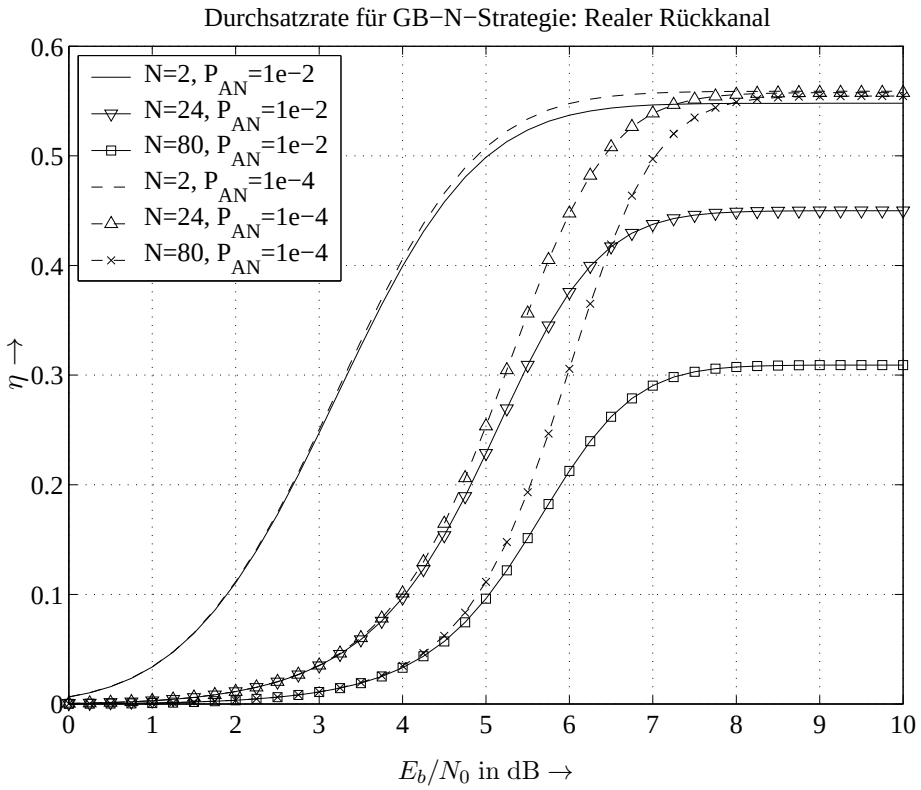

Bild 3.10: Vergleich verschiedener Fehlerwahrscheinlichkeiten P_{AN} für die GB-Strategie

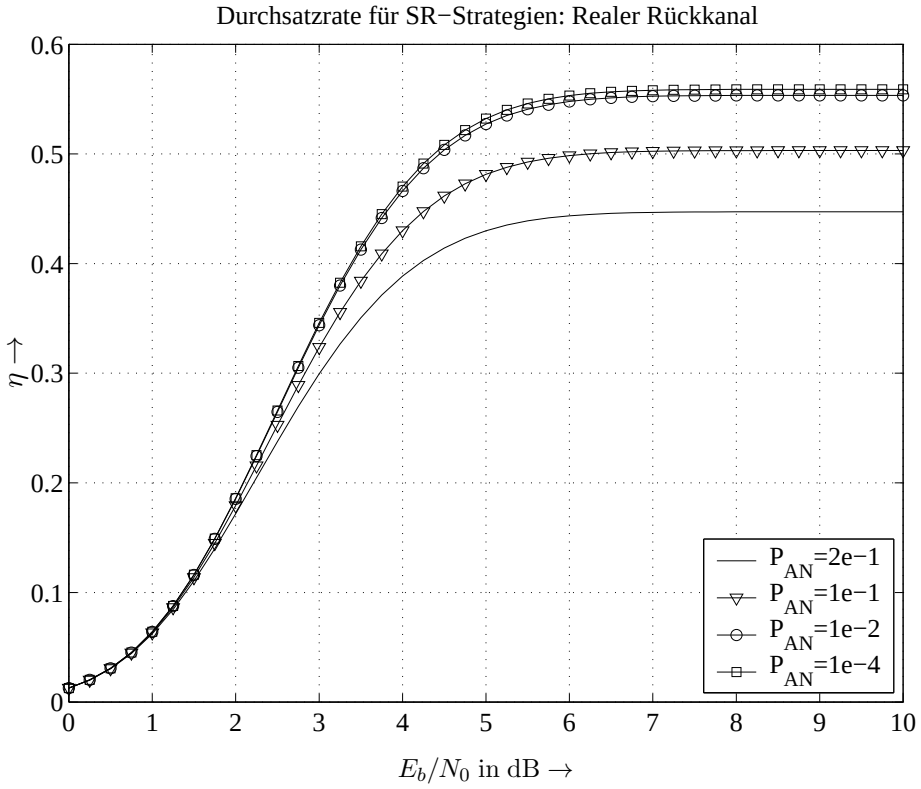

Bild 3.11: Vergleich verschiedener Fehlerwahrscheinlichkeiten P_{AN} für die SR-Strategie

3.5 Hybride FEC/ARQ-Systeme

Die beiden bisher betrachteten prinzipiellen Ansätze zur Kanalcodierung, die FEC- und die ARQ-Verfahren, besitzen, jedes für sich genommen, eine Reihe von Nachteilen. So kann man zwar mit reinen FEC-Verfahren

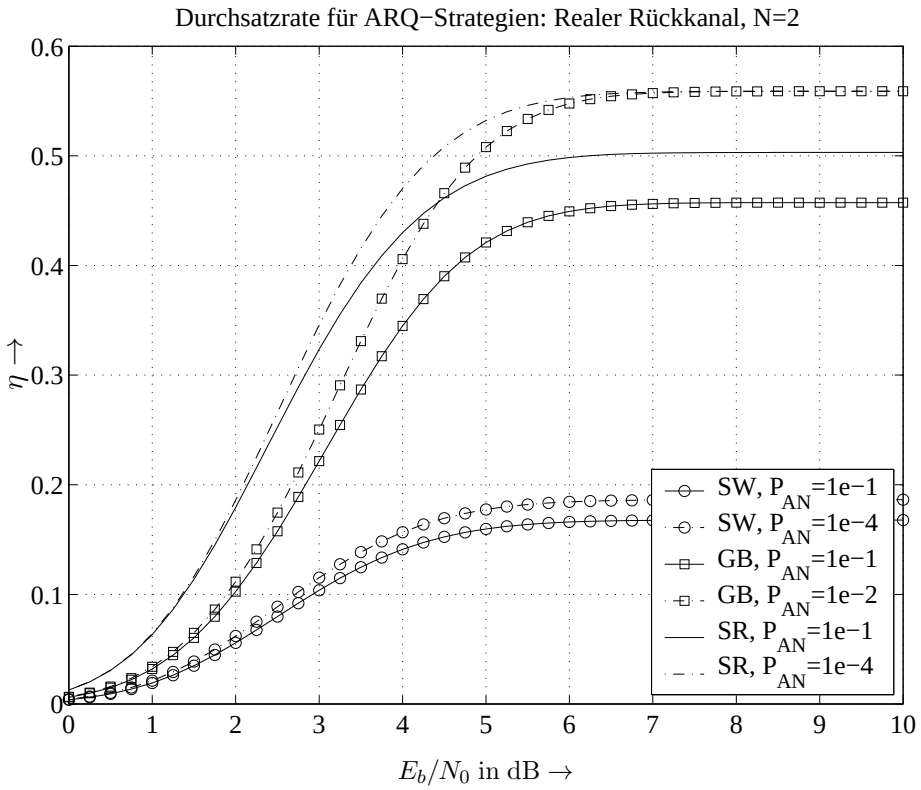

Bild 3.12: Vergleich der ARQ-Strategien für reale Rückkanäle mit $N = 2$

sehr niedrige Fehlerraten ab bestimmten Signal-Rausch-Abständen erzielen, für schlechtere Kanalbedingungen können sie jedoch keine fehlerfreie Übertragung garantieren. Andererseits fügen sie bei guter Kanalqualität viel mehr Redundanz zu, als eigentlich nötig wäre und reduzieren damit den möglichen Datendurchsatz.

Reine ARQ-Verfahren verhalten sich grundlegend anders. Mit einem entsprechend leistungsfähigen CRC-Code kann eine nahezu fehlerfreie Übertragung gewährleistet werden, allerdings auf Kosten einer u.U. gegen Null strebenden Durchsatzrate. Für gute Übertragungsbedingungen erreicht man hohe Durchsatzraten, da der fehlererkennende Code im Vergleich zu korrigierenden Codes weniger Redundanz benötigt. Wird der Kanal allerdings schlechter, verringern die ständigen Wiederholungen von fehlerhaften Blöcken die Effizienz dramatisch.

Es liegt also sehr nahe, die Vorteile beider Strategien gewinnbringend miteinander zu kombinieren. Bei guten bis mittleren Kanalbedingungen sorgt der FEC-Code für eine fast fehlerfreie Übertragung, der Nachteil des häufigen Wiederholens bei reinen ARQ-Techniken ist überwunden. Treten bei schlechteren Übertragungsbedingungen doch noch Fehler auf, sorgt die ARQ-Steuerung für entsprechende Wiederholungen. Beide Verfahren ergänzen sich also hervorragend zu einem sogenannten **hybriden FEC/ARQ-System**, dessen Struktur Bild 3.13 zeigt. Der zum ARQ-System gehörende fehlererkennende Code bildet den äußeren, der FEC-Code den inneren Code einer seriellen Verkettung.

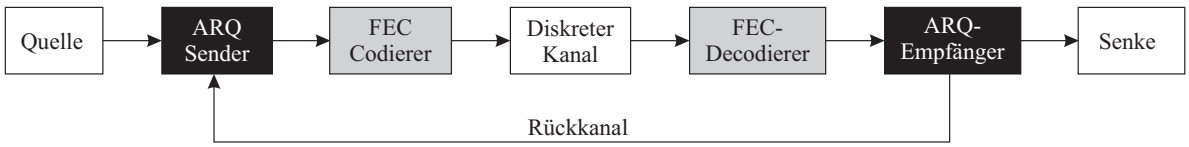

Bild 3.13: Struktur eines hybriden FEC-ARQ-Systems

3.5.1 Typ-I hybrides ARQ-System

Das sogenannte **Typ-I hybride ARQ-System** stellt die einfachste Art der Kombination von FEC und ARQ dar. Es besitzt eine gute Leistungsfähigkeit bei annähernd konstanten Übertragungsbedingungen, also quasi zeitinvarianten Kanälen. Entsprechend Bild 3.14 werden den Informationsbit zunächst die Prüfbit eines fehlererkennenden CRC-Codes angehängt (äußerer Code). Danach erfolgt die Codierung mit einem fehlerkorrigierenden FEC-Code, z.B. einem Faltungscode als inneren Code.

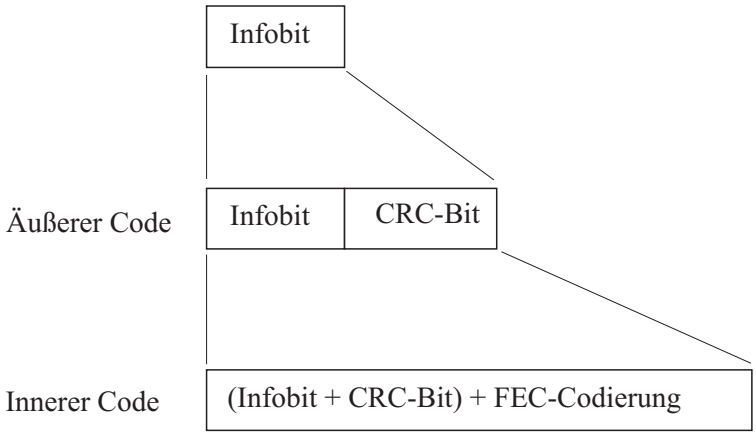

Bild 3.14: Codierung beim Typ-I hybriden ARQ-System

Die Funktionsweise ist mit der reiner ARQ-Verfahren identisch und in Bild 3.15 skizziert. Im fehlerfreien Fall erfolgt das Senden eines ACK-Signals, woraufhin der Sender den nächsten Block übermittelt. Wird ein Fehler detektiert, muss der betroffene Block wiederholt werden, inklusive der Redundanz des FEC-Codes. Der für bestimmte Kanäle erzielbare bessere Durchsatz im Vergleich zu reinen ARQ-Strategien wird durch die Möglichkeit der Fehlerkorrektur mit Hilfe des FEC-Codes erzielt. Ist dieser an den quasi konstanten Kanal angepaßt, kann eine nahezu optimale Durchsatzrate erreicht werden.

Allerdings besitzt das Typ-I hybride Verfahren weiterhin den gravierenden Nachteil reiner FEC-Systeme, dass keine ausreichende Adaption für zeitvariante Kanäle gelingt. Während guter Kanaleigenschaften wird durch den fehlerkorrigierenden Code mehr Redundanz als nötig eingebracht, die Effizienz verringert sich hier. Daher ist das Typ-I hybride Verfahren nicht für zeitvariante Kanäle geeignet. Für sie ist vielmehr eine Adaptivität auch des FEC-Codes erforderlich, auf die im folgenden Abschnitt näher eingegangen wird.

3.5.2 Hybrides ARQ-System mit ratenkompatiblen Faltungscodes

gesucht:	bessere Adaptivität der FEC-Komponente
	guter Kanal $\rightarrow$ wenig Redundanz, z.B. reines ARQ-System
	schlechter Kanal $\rightarrow$ viel Redundanz durch fehlerkorrigierenden Code
Lösung:	Redundanz des FEC-Codes wird nicht auf einmal übertragen, sondern sukzessive, je nachdem, wie oft Fehler aufgetreten sind und Wiederholungen angefordert werden. Wurde sofort fehlerfrei übertragen, reduziert sich hierdurch die übertragene Redundanz.

Für dieses ARQ-Prinzip eignen sich hervorragend punktierte Faltungscodes. Sie wurden für diese Anwendung erstmals von Hagenauer [Hag88] als **ratenkompatible punktierte Faltungscodes** (*Rate-Compatible Punctured Convolutional Codes*, RCPC-Codes) vorgestellt. Den Basiscode bildet ein gewöhnlicher Faltungscode mit der Coderate $R_c = 1/4$. Für ihn werden dann mehrere Punktierungsmatrizen $\mathbf{P}_l$ mit den Elementen $p_{i,j}(l)$

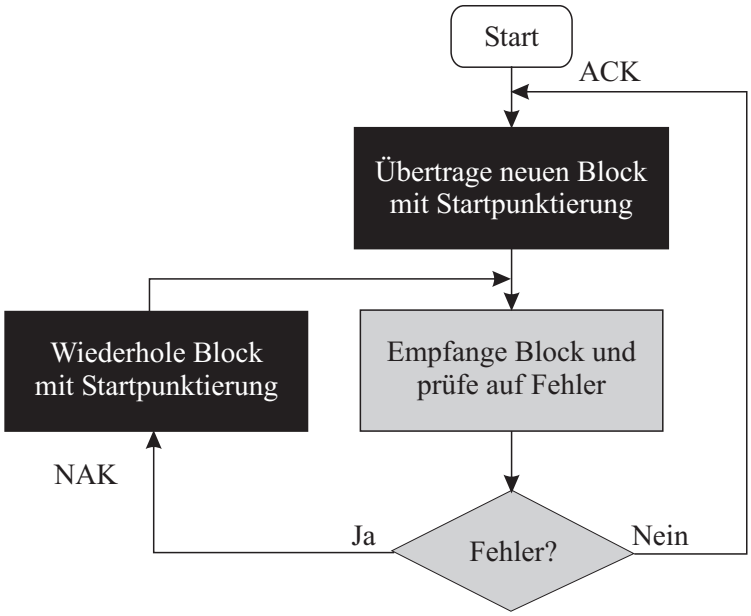

Bild 3.15: Protokollablauf beim Typ-I hybriden ARQ-System

entworfen, von denen jede eine bestimmte Coderate

$$R_c^{(l)} = \frac{L_P}{L_P + l} \tag{3.50}$$

repräsentiert, wobei L_P die Punktierungsperiode darstellt und der Parameter l im Intervall $1 \cdots (n-1)L_P$ liegt. Mit sinkendem Index l lassen sich dann Coderaten von $R_c = 1/4$ (keine Punktierung, $l = (n-1)L_P$) bis $R_c = 8/9$ ($l = 1$) einstellen.

Mit Hilfe der verschiedenen Punktierungsmatrizen kann zum einen Redundanz (Prüfbit) nachgesendet werden, wenn im Empfänger ein nicht korrigierbarer Fehler detektiert wurde. Weiterhin kann die Coderate schon direkt im Sender durch die Wahl einer geeigneten Matrix auf die momentanen Übertragungsbedingungen adaptiert werden. Hierzu kann z.B. eine Kanalzustandsschätzung im Empfänger dienen, deren Ergebnis über den Rückkanal dem Sender mitgeteilt wird.

Soll die Coderate gewechselt werden, ist auf die Bedingung der **Ratenkompatibilität** zu achten. Sie kann für einen Bezugsindex l_0 gemäß

$$p_{i,j}(l_0) = 1 \quad \Longrightarrow \quad p_{i,j}(l) = 1 \quad \forall\, l \geq l_0 \geq 1 \tag{3.51}$$

$$p_{i,j}(l_0) = 0 \quad \Longrightarrow \quad p_{i,j}(l) = 0 \quad \forall\, l \leq l_0 \leq (n-1)L_P - 1 \tag{3.52}$$

formuliert werden und besagt, dass bei einem Wechsel der Punktierungsmatrix von l zu $l+i$ (mehr Redundanz) alle bisher übertragenen Stellen auch weiterhin übertragen werden müssen. Es dürfen nur diejenigen Redundanzbit hinzukommen, die bisher punktiert wurden. Weiterhin dürfen bei einer Verringerung der Redundanz entsprechend Gl. (3.52) nur bisher übertragene Codestellen zusätzlich punktiert werden, die bisher punktierten Stellen werden auch weiterhin ausgeblendet. Die Ratenkompatibilität garantiert, dass beim Umschalten der Punktierungsmatrix keine Information verloren geht.

Die Realisierung des obigen Ansatzes ist in Bild 3.16 dargestellt. Im Sender werden die Informationsbit zunächst mit einem CRC-Code und dann mit einem FEC-Code niedriger Coderate (viel Redundanz) codiert. Die erste Übertragung beinhaltet aber nur einen Teil der FEC-Redundanz, der Rest wird in einem Puffer gespeichert. Stellt der Empfänger einen Fehler fest, speichert er den empfangenen Block ebenfalls und sendet ein NAK-Signal. Dies veranlaßt den Sender, eine erneute Übertragung zu initiieren. Es wird aber nicht

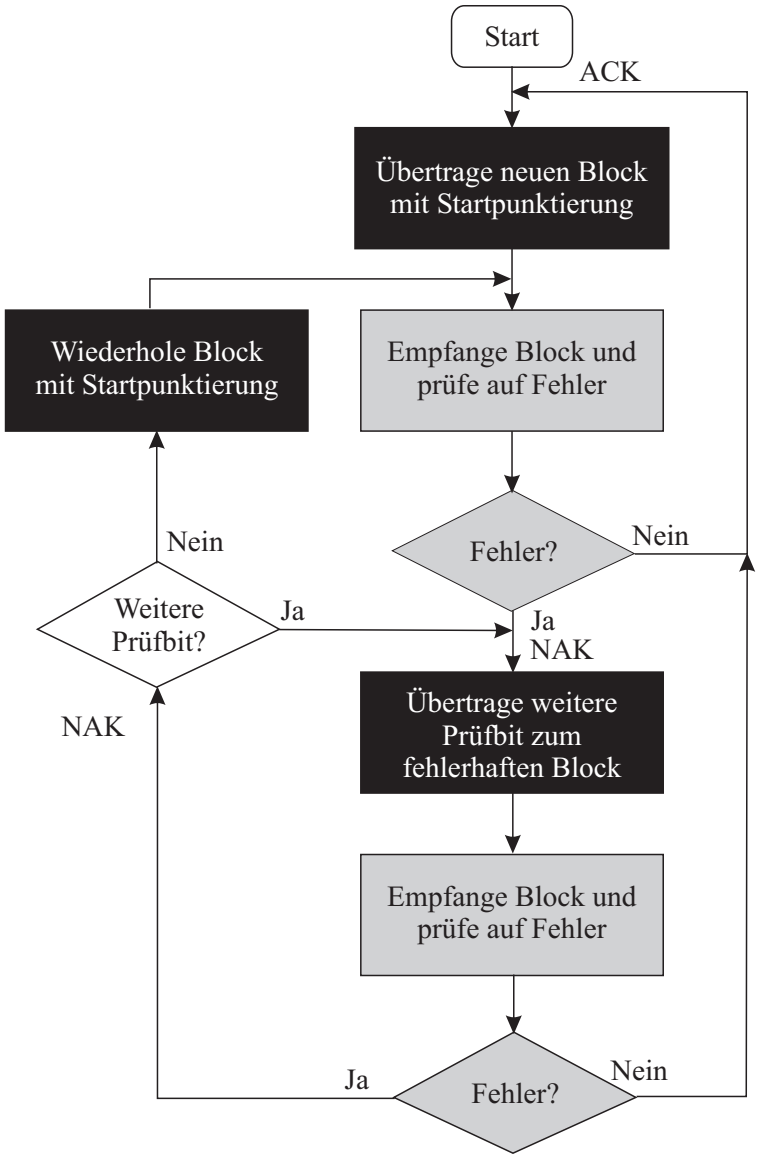

Bild 3.16: Protokollablauf beim hybriden ARQ-System mit RCPC-Codes

das ursprünglich gesendete Paket wiederholt, sondern vielmehr die bisher zurückgehaltenen Prüfbit der FEC-Codierung gesendet. Mit ihnen kann der Empfänger dann zusammen mit dem gepufferten, fehlerbehafteten Block einen erneuten Decodierversuch unternehmen. Die nun zur Verfügung stehende zusätzliche Redundanz soll jetzt zu einer fehlerfreien Decodierung führen.

Dabei werden in der Praxis nicht alle möglichen Coderaten tatsächlich genutzt. Es würde auch keinen Sinn machen, jedes Prüfbit einzelnen nachzusenden, da ein einzelnes zusätzliches Bit mit hoher Wahrscheinlichkeit nicht sofort zu einer erfolgreichen Fehlerkorrektur führt und somit mehrere Durchläufe erforderlich wären, von denen jeder die Verzögerungszeit erhöht. Vielmehr ist es sinnvoll, bestimmte Bündel zu definieren und diese im Fehlerfall zu übertragen. Dabei können beim Nachsenden der Prüfbit auch mehrere Bündel zusammengefaßt werden, um unnötige erfolglose Decodierversuche zu vermeiden. Wir müssen also einen Kompromiß zwischen möglichst geringer Redundanz und möglichst kleiner Verzögerungszeit suchen. Welche Strategie den erhofften Erfolg bringt, hängt entscheidend vom Übertragungskanal ab und kann nicht pauschal beantwortet werden.

Beispiel 1:
Basiscode mit $L_c = 5$ und $R_c = 1/4$

Generatorpolynome:

$$\begin{aligned}\mathbf{g}_0 &= 1 + D + D^4\\ \mathbf{g}_1 &= 1 + D^2 + D^3 + D^4\\ \mathbf{g}_2 &= 1 + D + D^2 + D^4\\ \mathbf{g}_3 &= 1 + D + D^3 + D^4\end{aligned}$$

Punktierungsmatrizen (Punktierungsperiode $L_P = 8 \longrightarrow$ 8 Spalten, Coderate $R_c = 1/4 \longrightarrow$ 4 Zeilen):

$$R_c = \frac{1}{4} \longrightarrow \mathbf{P}_0 = \begin{pmatrix} 1&1&1&1 & 1&1&1&1\\ 1&1&1&1 & 1&1&1&1\\ 1&1&1&1 & 1&1&1&1\\ 1&1&1&1 & 1&1&1&1 \end{pmatrix} \qquad R_c = \frac{4}{15} \longrightarrow \mathbf{P}_1 = \begin{pmatrix} 1&1&1&1 & 1&1&1&1\\ 1&1&1&1 & 1&1&1&1\\ 1&1&1&1 & 1&1&1&1\\ 1&1&1&0 & 1&1&1&0 \end{pmatrix}$$

$$R_c = \frac{2}{7} \longrightarrow \mathbf{P}_2 = \begin{pmatrix} 1&1&1&1 & 1&1&1&1\\ 1&1&1&1 & 1&1&1&1\\ 1&1&1&1 & 1&1&1&1\\ 1&0&1&0 & 1&0&1&0 \end{pmatrix} \qquad R_c = \frac{4}{13} \longrightarrow \mathbf{P}_3 = \begin{pmatrix} 1&1&1&1 & 1&1&1&1\\ 1&1&1&1 & 1&1&1&1\\ 1&1&1&1 & 1&1&1&1\\ 1&0&0&0 & 1&0&0&0 \end{pmatrix}$$

$$R_c = \frac{1}{3} \longrightarrow \mathbf{P}_4 = \begin{pmatrix} 1&1&1&1 & 1&1&1&1\\ 1&1&1&1 & 1&1&1&1\\ 1&1&1&1 & 1&1&1&1\\ 0&0&0&0 & 0&0&0&0 \end{pmatrix} \qquad R_c = \frac{4}{11} \longrightarrow \mathbf{P}_5 = \begin{pmatrix} 1&1&1&1 & 1&1&1&1\\ 1&1&1&1 & 1&1&1&1\\ 1&1&1&0 & 1&1&1&0\\ 0&0&0&0 & 0&0&0&0 \end{pmatrix}$$

$$R_c = \frac{4}{10} \longrightarrow \mathbf{P}_6 = \begin{pmatrix} 1&1&1&1 & 1&1&1&1\\ 1&1&1&1 & 1&1&1&1\\ 1&1&0&0 & 1&1&0&0\\ 0&0&0&0 & 0&0&0&0 \end{pmatrix} \qquad R_c = \frac{4}{9} \longrightarrow \mathbf{P}_7 = \begin{pmatrix} 1&1&1&1 & 1&1&1&1\\ 1&1&1&1 & 1&1&1&1\\ 1&0&0&0 & 1&0&0&0\\ 0&0&0&0 & 0&0&0&0 \end{pmatrix}$$

$$R_c = \frac{1}{2} \longrightarrow \mathbf{P}_8 = \begin{pmatrix} 1&1&1&1 & 1&1&1&1\\ 1&1&1&1 & 1&1&1&1\\ 0&0&0&0 & 0&0&0&0\\ 0&0&0&0 & 0&0&0&0 \end{pmatrix} \qquad R_c = \frac{4}{7} \longrightarrow \mathbf{P}_9 = \begin{pmatrix} 1&1&1&1 & 1&1&1&1\\ 1&1&1&0 & 1&1&1&0\\ 0&0&0&0 & 0&0&0&0\\ 0&0&0&0 & 0&0&0&0 \end{pmatrix}$$

$$R_c = \frac{2}{3} \longrightarrow \mathbf{P}_{10} = \begin{pmatrix} 1&1&1&1 & 1&1&1&1\\ 1&0&1&0 & 1&0&1&0\\ 0&0&0&0 & 0&0&0&0\\ 0&0&0&0 & 0&0&0&0 \end{pmatrix} \qquad R_c = \frac{4}{5} \longrightarrow \mathbf{P}_{11} = \begin{pmatrix} 1&1&1&1 & 1&1&1&1\\ 1&0&0&0 & 1&0&0&0\\ 0&0&0&0 & 0&0&0&0\\ 0&0&0&0 & 0&0&0&0 \end{pmatrix}$$

$$R_c = \frac{8}{9} \longrightarrow \mathbf{P}_{12} = \begin{pmatrix} 1&1&1&1 & 0&1&1&1\\ 1&0&0&0 & 1&0&0&0\\ 0&0&0&0 & 0&0&0&0\\ 0&0&0&0 & 0&0&0&0 \end{pmatrix}$$

Beispiel 2:
Basiscode mit $L_c = 7$ und $R_c = 1/3$

Generatorpolynome:

$$\begin{aligned}\mathbf{g}_0 &= 1 + D + D^3D^4 + D^6\\ \mathbf{g}_1 &= 1 + D^3 + D^4 + D^5 + D^6\\ \mathbf{g}_2 &= 1 + D^2 + D^5 + D^6\end{aligned}$$

Punktierungsmatrizen (Punktierungsperiode $L_P = 8 \longrightarrow$ 8 Spalten, Coderate $R_c = 1/3 \longrightarrow$ 3 Zeilen):

$$R_c = \frac{1}{3} \longrightarrow \mathbf{P}_0 = \begin{pmatrix} 1\,1\,1\,1 & 1\,1\,1\,1 \\ 1\,1\,1\,1 & 1\,1\,1\,1 \\ 1\,1\,1\,1 & 1\,1\,1\,1 \end{pmatrix} \qquad R_c = \frac{4}{11} \longrightarrow \mathbf{P}_1 = \begin{pmatrix} 1\,1\,1\,1 & 1\,1\,1\,1 \\ 1\,1\,1\,1 & 1\,1\,1\,1 \\ 1\,1\,1\,0 & 1\,1\,1\,0 \end{pmatrix}$$

$$R_c = \frac{4}{10} \longrightarrow \mathbf{P}_2 = \begin{pmatrix} 1\,1\,1\,1 & 1\,1\,1\,1 \\ 1\,1\,1\,1 & 1\,1\,1\,1 \\ 1\,1\,0\,0 & 1\,1\,0\,0 \end{pmatrix} \qquad R_c = \frac{4}{9} \longrightarrow \mathbf{P}_3 = \begin{pmatrix} 1\,1\,1\,1 & 1\,1\,1\,1 \\ 1\,1\,1\,1 & 1\,1\,1\,1 \\ 1\,0\,0\,0 & 1\,0\,0\,0 \end{pmatrix}$$

$$R_c = \frac{1}{2} \longrightarrow \mathbf{P}_4 = \begin{pmatrix} 1\,1\,1\,1 & 1\,1\,1\,1 \\ 1\,1\,1\,1 & 1\,1\,1\,1 \\ 0\,0\,0\,0 & 0\,0\,0\,0 \end{pmatrix} \qquad R_c = \frac{4}{7} \longrightarrow \mathbf{P}_5 = \begin{pmatrix} 1\,1\,1\,1 & 1\,1\,1\,1 \\ 1\,1\,1\,0 & 1\,1\,1\,0 \\ 0\,0\,0\,0 & 0\,0\,0\,0 \end{pmatrix}$$

$$R_c = \frac{2}{3} \longrightarrow \mathbf{P}_6 = \begin{pmatrix} 1\,1\,1\,1 & 1\,1\,1\,1 \\ 1\,0\,1\,0 & 1\,0\,1\,0 \\ 0\,0\,0\,0 & 0\,0\,0\,0 \end{pmatrix} \qquad R_c = \frac{4}{5} \longrightarrow \mathbf{P}_7 = \begin{pmatrix} 1\,1\,1\,1 & 1\,1\,1\,1 \\ 1\,0\,0\,0 & 1\,0\,0\,0 \\ 0\,0\,0\,0 & 0\,0\,0\,0 \end{pmatrix}$$

$$R_c = \frac{8}{9} \longrightarrow \mathbf{P}_8 = \begin{pmatrix} 1\,1\,1\,1 & 0\,1\,1\,1 \\ 1\,0\,0\,0 & 1\,0\,0\,0 \\ 0\,0\,0\,0 & 0\,0\,0\,0 \end{pmatrix}$$

3.5.3 Typ-II hybrides System

Ein Nachteil des oben beschriebenen Systems besteht darin, dass der erste gesendete Block im Fehlerfall immer wieder für die Decodierung herangezogen wird. Zwar können die nachträglich übertragenen Prüfbit häufig zur fehlerfreien Decodierung führen, wenn das erste Paket aber sehr stark gestört ist, bleiben unter Umständen aber alle Decodierversuche erfolglos. Diesen Nachteil behebt das **Typ-II hybride ARQ-System**. Die prinzipielle Vorgehensweise ähnelt der des vorigen Verfahrens und ist in Bild 3.17 illustriert.

Auch beim Typ-II hybriden ARQ-Verfahren wird im Sender zunächst Redundanz zurückgehalten, die dann bei Eintreten eines Übertragungsfehlers nachgeliefert wird. Voraussetzung ist hier allerdings der Einsatz invertierbarer Codes, d.h. sind die Prüfbit eines Codewortes korrekt empfangen worden, kann eindeutig auf die Informationsbit zurückgeschlossen werden. Dies hat den Vorteil, dass bei zeitvarianten Kanälen die nachgesendete Redundanz unter Umständen viel besser übertragen wurde als der Originalblock und dann allein, also ohne die Nutzung der gepufferten ersten Version, fehlerfrei decodiert werden kann. In diesem Fall ist eine gemeinsame Decodierung beider Blöcke überflüssig, sie wäre sogar nachteilig. Konventionelle Blockcodes erfüllen die Bedingung der Invertierbarkeit nicht immer, da in der Regel weniger Prüfbit als Informationsbit vorhanden sind und somit eine bijektive Abbildung zwischen Informations- und Prüfbit ausgeschlossen ist.

Verfahren nach Lin/Yu
Bei der Realisierung nach Lin und Yu wird der Informationsvektor $\mathbf{u}$ zunächst mit einem fehlererkennenden Code C_1 codiert, es werden also Prüfbit angehängt. Das Codewort lautet ($\mathbf{c}_1 = \mathbf{u}, \mathbf{q}_1$). Zusätzlich erfolgt die Codierung mit einem zweiten, systematischen, fehlerkorrigierenden und invertierbaren Code C_2 (s. Bild 3.18), dessen Prüfteil $\mathbf{p}_2$ allerdings nicht gesendet, sondern im Sender gepuffert wird. Zuerst wird $\mathbf{c}_1$ gesendet. Tritt während der Übertragung ein Fehler auf, so wird der empfangene Block $(\mathbf{u}, \mathbf{q}_1)$ im Empfänger gespeichert und ein NAK-Signal gesendet. Dies veranlaßt den Sender, die Prüfbit $\mathbf{p}_2$ des Codes C_2 ebenfalls mit dem fehlererkennenden Code C_1 zu codieren $\mathbf{c}_2 = (\mathbf{p}_2, \mathbf{q}_2)$ und zu übertragen.

Es ist zu beachten, dass bei der ersten Übertragung nur die Informationsbit und einige wenige Prüfbit des fehlererkennenden Codes C_1 gesendet wurden, die Redundanz ist also sehr gering. Im Fehlerfall werden die

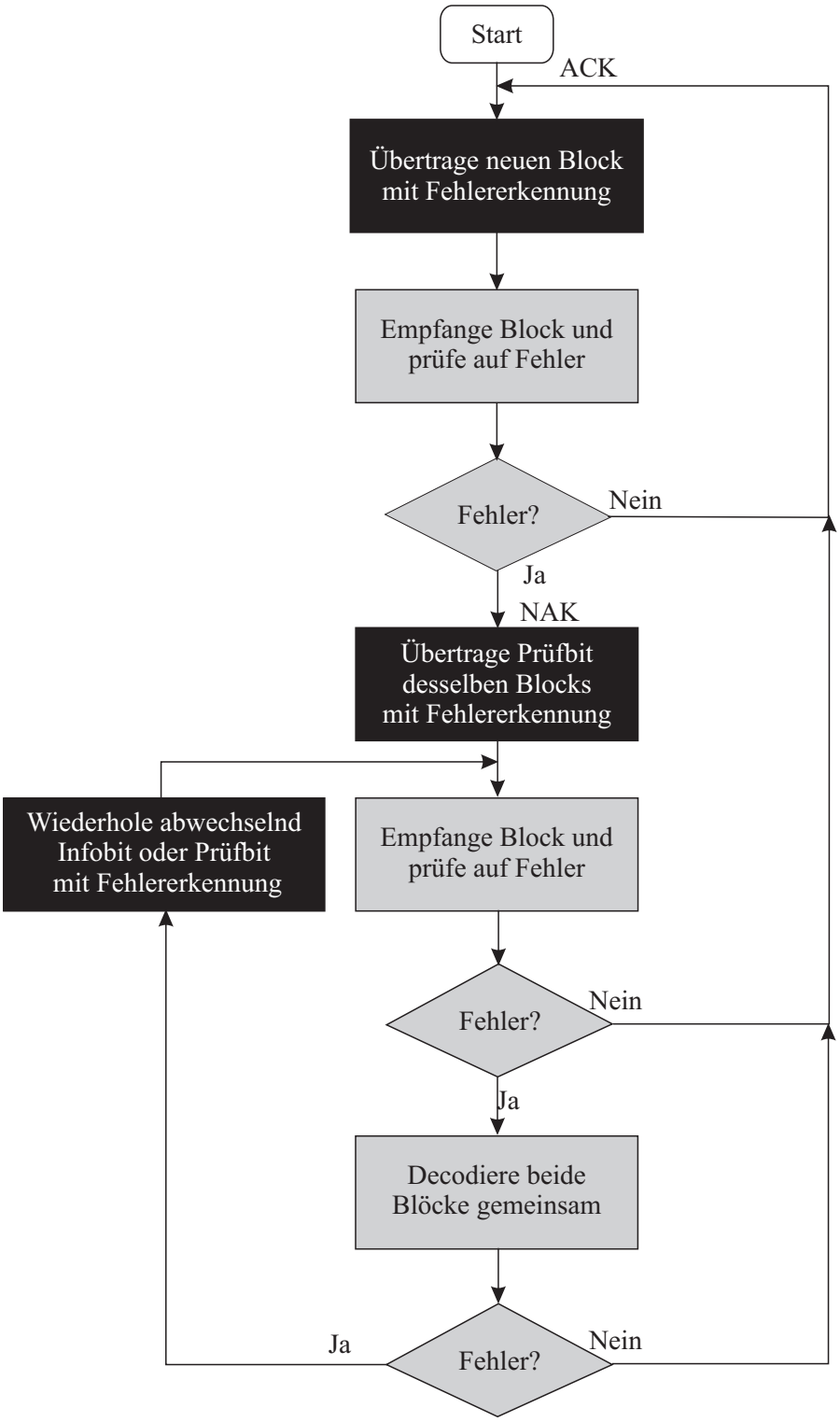

Bild 3.17: Protokollablauf beim Typ-II hybriden ARQ-Systems

Informationsbit dann nicht mehr übertragen, sondern die Prüfbit des fehlerkorrigierenden und invertierbaren Codes C_2 (zusammen mit Prüfbit $\mathbf{q}_2$).

Im Empfänger werden nun zunächst die empfangenen Prüfbit $\mathbf{p}_2$ anhand der Checksumme $\mathbf{q}_2$ auf Fehlerfreiheit überprüft. Ist dies der Fall, so können aufgrund der Invertierbarkeit von Code C_2 die Informationsbit $\mathbf{u}$ direkt aus $\mathbf{p}_2$ berechnet werden. Wird dagegen ein Fehler detektiert, so erfolgt die gemeinsame Decodierung von $(\mathbf{u}, \mathbf{p}_2)$ des Codes C_2. Ist diese auch erfolglos, werden die Informationsbit zusammen mit $\mathbf{q}_1$ erneut übertragen,

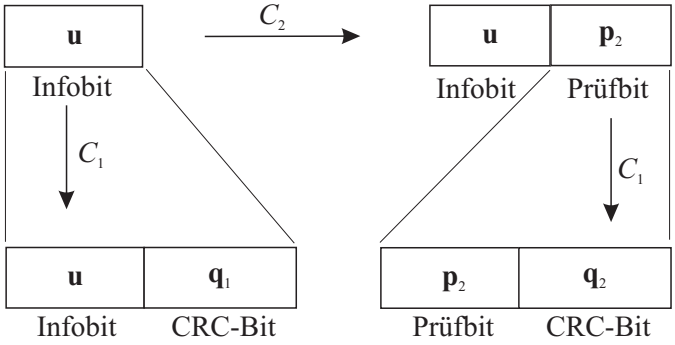

Bild 3.18: Codierung beim Typ-II hybriden ARQ-System

zunächst allein decodiert und im Fehlerfall erneut zusammen mit $\mathbf{p}_2$ decodiert usw.

Verfahren nach Lugand/Costello
Eine zweite, sehr ähnliche Variante des Typ-II hybriden ARQ-Verfahrens wurde von Lugand und Costello vorgestellt. Hier wird als fehlerkorrigierender Code C_2 ein halbratiger Faltungscode mit den Generatorpolynomen $g_0(D)$ und $g_1(D)$ eingesetzt. Da die Schieberegisterstruktur eines nicht-rekursiven Faltungscodes als FIR-Filter interpretiert werden kann, ist der Code durch ein IIR-Filter einfach invertierbar. Der CRC-Code besitze das Generatorpolynom $g(D)$.

Die Vorgehensweise ist nun die gleiche wie beim Lin/Yu-Verfahren. Zuerst wird das Codewort $c_0(D) = u(D) \cdot g_0(D) \cdot g(D)$ berechnet und übertragen. Dies entspricht einer Hälfte des halbratigen Faltungscodes, es ist also noch keine Redundanz durch C_2 zugefügt worden. Die Informationsbit sind durch die Faltung mit $g_0(D)$ lediglich miteinander kombiniert worden. Tritt kein Fehler auf, so kann aus der Sequenz $u(D) \cdot g_0(D)$ mit Hilfe des IIR-Filters die gesuchte Informationsfolge $u(D)$ geschätzt werden. Im Fehlerfall wird dagegen im Sender auch das zweite Generatorpolynom des Faltungscodes genutzt. Wir bilden $c_1(D) = u(D) \cdot g_1(D) \cdot g(D)$ und übertragen es. Auch $c_1(D)$ kann einzeln decodiert werden.

Ist auch dieser Versuch gescheitert, werden $(c_0(D), c_1(D))$ gemeinsam mit dem Viterbi Algorithmus decodiert. Schlägt auch diese Decodierung fehl, ist die Sequenz $c_0(D)$ zu wiederholen, zunächst einzeln zu decodieren, dann gegebenenfalls gemeinsam mit $c_1(D)$, usw.

3.6 Typ-III hybrides System

Das Typ-II hybride ARQ-System hat gegenüber dem Typ-I hybriden Ansatz den Vorteil, dass jede Teilübertragung für sich einzeln decodierbar ist und somit nicht unter den *Altlasten* vorangegangener Versuche leidet. Zudem besitzt es weiterhin den Vorteil, mehrere, durch unterschiedliche Übertragungsbedingungen gestörte Empfangssignale miteinander zu kombinieren und somit Diversität auszunutzen.

Diese Idee greift nun das letzte in diesem Kapitel vorgestellte Verfahren noch einmal auf, nämlich das **Typ-III hybride Verfahren** [Kal95]. Es verwendet die schon erwähnten ratenkompatiblen punktierten Faltungscodes und nutzt gleichzeitig die Vorteile der Typ-II hybriden Strategien. Dies wird durch die sogenannte komplementäre Punktierung erreicht.

Komplementäre Punktierung bedeutet, dass ein niederratiger Faltungscode genommen wird, der mit verschiedenen Punktierungsmatrizen $\mathbf{P}_i$ punktiert wird, wobei sich die $\mathbf{P}_i$ derart ergänzen, dass jedes Bit mindestens einmal übertragen wird. Jede Punktierungsmatrix einzeln gesehen sorgt dafür, dass nur wenig Redundanz je Übertragung gesendet wird, die zugehörige Sequenz ist einzeln decodierbar.

Nachgesendete Versionen anderer $\mathbf{P}_i$ können im Fehlerfall ohne Probleme mit ihren Vorgängern kombiniert

und mit einem einzigen Viterbi Algorithmus decodiert werden. Dabei kann es durchaus vorkommen, dass einzelne Symbole mehrfach übertragen wurden. Lediglich das komplette Ausblenden bestimmter Bit durch alle Punktierungsmatrizen ist zu vermeiden.

Komplementäre Punktierungsmatrizen können durch zyklisches Verschieben der Spalten von $\mathbf{P}_i$ gebildet werden.

Beispiel

$$\mathbf{P}_1 = \begin{pmatrix} 1\,1\,0\,1 & 1\,1\,0\,1 \\ 1\,0\,1\,1 & 1\,0\,1\,1 \\ 1\,1\,1\,1 & 1\,1\,1\,1 \\ 0\,1\,1\,0 & 0\,1\,1\,0 \end{pmatrix}$$

$$\mathbf{P}_2 = \begin{pmatrix} 0\,1\,1\,1 & 0\,1\,1\,1 \\ 1\,1\,1\,0 & 1\,1\,1\,0 \\ 1\,1\,1\,1 & 1\,1\,1\,1 \\ 1\,0\,0\,1 & 1\,0\,0\,1 \end{pmatrix}$$

$$\mathbf{P}_1 + \mathbf{P}_2 = \begin{pmatrix} 1\,2\,1\,2 & 1\,2\,1\,2 \\ 2\,1\,2\,1 & 2\,1\,2\,1 \\ 2\,2\,2\,2 & 2\,2\,2\,2 \\ 1\,1\,1\,1 & 1\,1\,1\,1 \end{pmatrix}$$

Es ist zu erkennen, dass die Punktierungsmatrizen $\mathbf{P}_i$ die primitive Periode $L_P = 4$ besitzen. Sie haben jedoch 8 Spalten, damit sie durch zyklisches Verschieben der Spalten um $L_P/2 = 2$ Positionen ineinander überführt werden können.

Für dieses einfache Beispiel würden wir ein System erhalten, dass bei jeder Übertragung einen drittelratigen Faltungscode verwendet, insgesamt wird bei 2 Übertragungen eine Gesamtcoderate von $R_c = 1/6$ erzielt, was höher ist als zweimal den viertelratigen Faltungscode zu verwenden.

Literaturverzeichnis

[BDMS91] E. Biglieri, D. Divsalar, P. McLane, and M. Simon. *Introduction to Trellis-Coded Modulation with Applications*. Macmillan Publishing Company, New York, 1991.

[Bos98] M. Bossert. *Kanalcodierung*. Teubner, Stuttgart, 1998.

[For66] G. D. Forney. *Concatenated Codes*. The M.I.T. Press, Cambridge, Massachusetts, 1966.

[Fri96] Bernd Friedrichs. *Kanalcodierung*. Springer Verlag, Berlin, 1996.

[Hag88] J. Hagenauer. Rate-Coompatible Punctured Convolutional Codes (RCPC-Codes and their Applications. *IEEE Transactions on Communications*, 36(4):389–400, April 1988.

[Hag96] J. Hagenauer. Source-controlled channel decoding. *IEEE Tran. on Communications*, 43(9):2449–2457, September 1996.

[Kal95] S. Kallel. Complementary-Punctured Convolutional (CPC) Codes and their Applications. *IEEE Transactions on Communications*, 43(6):2005–2009, June 1995.

[Kam04] K. D. Kammeyer. *Nachrichtenübertragung*. Teubner, Stuttgart, 2004.

[Off96] E. Offer. *Decodierung mit Qualitätsinformation bei verketteten Codiersystemen*. PhD thesis, Deutsche Forschungsanstalt für Luft- und Raumfahrt (DLR), VDI Fortschrittsberichte, Juni 1996.

[Ung82] G. Ungerböck. Channel Coding with Multilevel Phase Signals. *IEEE Transactions on Information Theory*, IT-28:56–67, January 1982.

[Ung87] G. Ungerböck. Trellis-Coded Modulation with Redundant Signal Sets. Part I: Introduction; Part II: State of the Art. *IEEE Communications Magazine*, 25(2):5–21, February 1987.

[VWZP89] A.J. Viterbi, J.K. Wolf, E. Zehavi, and R. Padovani. A Pragmatic Approach to Trellis-Coded Modulation. *IEEE Communications Magazine*, 27(6):11–19, July 1989.