1. Vorwärtsfehlerkorrektur

In diesem Kapitel wird ein Verfahren vorgestellt, mit dem Paketverluste durch *Vorwärtsfehlerkorrektur* ("Forward Error Correction") ausgeglichen werden. Vorwärtsfehlerkorrekturverfahren werden meistens von allgemeineren Fehlerkorrekturverfahren abgeleitet, die auch Sendefehler korrigieren können. In unserem Fall (Senden über IP) werden Sendefehler durch die unterliegenden Netzwerkschichten schon behandelt, so dass wir uns nur noch um Paketverluste kümmern müssen. Hier kommen sogenannte *Block Erasure Codes* zum Einsatz. Wir setzen das von Rizzo in [**?**] beschriebene Verfahren ein, welches sich effizient in Software implementieren lässt.

1.1 Ein lineares, systematisches Verfahren

1.1.1 Lineare, systematische Codes

Bei Fehlerkorrekturverfahren, die Paketverluste ausgleichen, werden k Quelldatenpakete in n ($n > k$) kodiert, und n Pakete anschliessend versendet. k empfangene Pakete sind ausreichend, um die Quelldaten wieder herzustellen. Das Verfahren von Rizzo lässt sich in die Klasse der *linearen Fehlerkorrekturcodes* einordnen. Dabei werden die k Quellpakete ($\overrightarrow{x} = x_0 \dots x_{k-1}$) an eine $n \times k$ Kodierungsmatrix G multipliziert, um den zu sendenden Paketvektor $\overrightarrow{y}$ zu erhalten:

$$\overrightarrow{y} = G\overrightarrow{x}.$$

Damit die empfangenen Pakete aus $\overrightarrow{y}$ wieder zu den Quellpaketen konvertiert werden können muss jede Untermatrix von G, die k Zeilen hat, invertierbar sein. Die Matrix G wird auch *Generatormatrix* genannt.

Weiterhin kann ein Fehlerkorrekturverfahren *systematisch* sein, wenn die unveränderten Quelldaten in dem Ergebnisvektor zu finden sind, also

$$x_0 \dots x_{k-1} \in \overrightarrow{y}.$$

Das lässt sich am einfachsten realisieren, wenn die Identitätsmatrix I_k eine Untermatrix von G ist. Ein systematisches Fehlerkorrektursverfahren hat den Vorteil, dass bei geringem Paketverlust die Quelldaten mit geringem Berechnungsaufwand wiederhergestellt werden können.

Wenn k Pakete aus $\overrightarrow{y}$ ($\overrightarrow{y'}$) empfangen werden können die k Quellpakete wiederhergestellt werden, indem das Inverse der G' Matrix, die aus den k Gleichungen zu den Paketen aus $\overrightarrow{y}$ besteht, an $\overrightarrow{y'}$ multipliziert wird:

$$\overrightarrow{y'} = G'\overrightarrow{x} \leftrightarrow \overrightarrow{x} = G'^{-1}\overrightarrow{y'}.$$

Um die Matrix G' zu berechnen muss die Sequenznummer der gesendeten Pakete bekannt sein, damit es nicht zu fatalen Umordnungen kommt. Meistens wird diese Information schon durch den unterliegenden Kommunikationskanal zur Verfügung gestellt (in unserem Fall durch ein Feld in dem Paketheader der gesendeten Datenpakete).

1.1.2 Galoisfeldarithmetik

Wenn die Quelldaten (also $x_i, 0 \leq i < n$) mit b Bits kodiert werden, und die Elemente der Kodierungsmatrix G mit b' Bits, müssen die Ergebnisse der Kodierungsmultiplikation $G\overrightarrow{x}$ mit $b + b'$ kodiert werden, um die Genauigkeit zu erhalten. Der zusätzliche Datenaufwand ist bei dem Einsatzgebiet des Verfahrens nicht vertretbar (Verdoppelung der zu sendenden Daten).

Wenn jedoch in einem *endlichen Körper* gerechnet wird, sind die Ausgangswerte mit derselben Anzahl Bits kodierbar wie die Eingangselemente (ein endlicher Körper ist unter Addition und Multiplikation geschlossen). In dem Verfahren von Rizzo werden Galoiskörper (auch Erweiterungskörper) eingesetzt.

Primkörper sind endliche Körper mit p Elementen, wo p Primzahl ist, und werden $GF(p)$ notiert. Addition und Multiplikation in einem Primkörper sind Addition und Multiplikation modulo p. Da Primkörper sich nur ineffizient implementieren lassen (mit Ausnahme von $p = 2$ gehen immer Bits bei der Kodierung von Elemente in $GF(p)$ verloren) werden *Erweiterungskörper* eingesetzt, die p^r Elemente haben und $GF(p^r)$ notiert werden. Elemente aus $GF(p^r)$ kann man auch als Polynome des Grades $r - 1$ über $GF(p)$ auffassen. Addition ist die koeffizientenweise Addition modulo p, Multiplikation die Polynommultiplikation modulo einem Minimalpolynom in $GF(p^r)$ (ohne Teiler in $GF(p^r)$). In jedem Galoiskörper G existiert ein *Generatorelement* α, das mit sich selbst multipliziert alle nicht-null Elemente aus G spannt (also $\alpha^0, \alpha^1, \ldots, \alpha^{p^r}$). Jedem Element $x \in G$ kann man also den α-Logarithmus k, so dass $\alpha^k = x$, zuweisen.

Die Operationen in $GF(p^r)$ lassen sich für $p = 2$ extrem effizient implementieren, da Addition sich durch XOR und Multiplikation sich durch Logarithmuslookuptabellen implementieren lassen.

1.1.3 Erstellen der Generatormatrix G

Die $n \times k$ Generatormatrix G muss 2 Bedingungen erfüllen:

- Jede Untermatrix von G aus k Zeilen muss invertierbar sein, damit die Dekodierungsmatrix G'^{-1} auch berechnet werden kann.
- Die Matrix G muss die $k \times k$ Identitätsmatrix enthalten, damit das Fehlerkorrekturverfahren auch systematisch ist.

Um eine solche Matrix zu generieren wird zuerst eine $k \times n$ Vandermonde Matrix erstellt, so dass die erste Zeile Potenzen von 0 beinhaltet. Die obere $k \times k$ Matrix wird invertiert und an die untere $k \times (n - k)$ Matrix multipliziert. Die obere $k \times k$ Matrix wird dann mit der $k \times k$ Identitätsmatrix gefüllt. Damit ist gewährleistet, dass die Generatormatrix korrekt und systematisch ist.

1.1.4 Kodieren von Quellpaketen

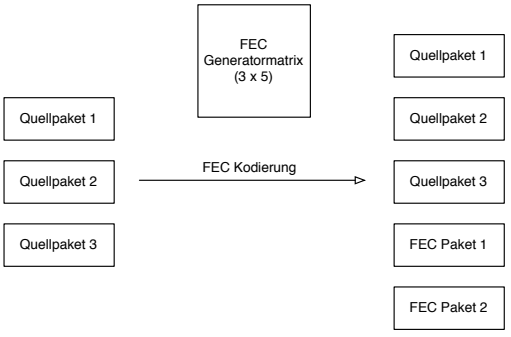

Abbildung 1.1: Kodierung von Quelldaten

Zum Kodieren des i-ten Zielpaketes werden die k Quellpakete mit der i-ten Zeilen der Generatormatrix multipliziert und dann summiert. Dazu wird das j-te Quellpakete mit der j-ten Komponente der i-ten Zeilen multipliziert, so dass bei einer systematischen Matrix, die die Einheitsmatrix enthält, die ersten k Zielpakete dieselben Daten enthalten wie die ersten k Quellpakete (siehe Abb. **??**).

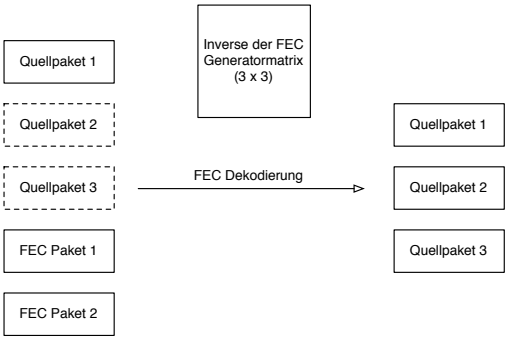

Abbildung 1.2: Dekodieren von FEC-kodierten Daten

1.1.5 Dekodieren von empfangenen Paketen

Zum Zurückgewinnen des i-ten Quellpaketes aus k empfangenen Paketen müssen die empfangenen Pakete mithilfe einer Untermatrix der Generatormatrix dekodiert werden. Diese Dekodiermatrix wird berechnet, indem die Zeilen die den Indexen der empfangenen Matrix entsprechen aus der Generatormatrix in eine temporäre $k \times k$ Matrix kopiert werden, und diese invertiert wird. Werden jetzt analog zum Kodierverfahren die empfangenen Daten an diese Matrix multipliziert, gewinnt man die Quelldaten wieder (siehe Abb. **??**).

1.2 Einsatz im Bereich MP3 Streaming

Genau wie bei dem Verfahren in RFC 3119 sollten bei redundant kodierten MP3-Daten ADUs verwendet werden, so dass bei einem Paketverlust der nicht aufgehoben werden kann nicht gleich mehrere MP3-Frames verloren gehen. Die eigentliche Fehlerkorrekturkodierung erfolgt also nach dem Konvertieren der Eingangsframe in ADUs. Dazu werden die ADUs in Gruppen von k ADUs gesammelt, und aus denen dann n Pakete kodiert, die dann versendet werden können. Diese Gruppen werden FEC-Gruppen genannt. Je nach Qualität des unterliegenden Kommunikationsmedium können die Grössen (k) dieser Gruppen und die Anzahl der daraus kodierten Pakete (n) varieert werden, um ein gutes Verhältnis zwischen benutzter Bandbreite und effektiver Fehlerkorrektur zu erreichen.

1.2.1 Verarbeitung von ADUs verschiedener Länge

Jetzt tritt allerdings ein anderes Problem auf. Im Gegensatz zu MP3-Frames können ADUs verschieden lang sein (und sie es in der Praxis auch immer). Das Verfahren von Rizzo kodiert allerdings nur gleichlange Datenpakete. Man könnte die ADUs in einen langen Datenpuffer zusammenhängen, und diesen Puffer dann in gleichgroße Pakete unterteilen, die redundant kodiert werden. Dieses Verfahren hat allerdings den Nachteil, daß bei grösseren Paketverlusten, bei denen die Quelldaten aus den empfangenen Daten nicht mehr hergestellt werden können die ADUs der kompletten FEC-Gruppe verloren gehen. Da wir jedoch ein systematisches FEC-Verfahren benutzen, können die ADUs mit Nullen auf die Größe l des größten ADUs aufgefüllt werden, um somit n Pakete der Länge l zu kodieren, die ersten k Pakete brauchen aber jeweils nur der Länge des entsprechenden ADUs zu sein, da sie auf der Empfängerseite wieder mit Nullen aufgefüllt werden können. So können bei größeren Paketverlusten immer noch die empfangenen Pakete unter den k ersten FEC-Paketen als MP3 ADUs benutzt werden (siehe Abb. **??**).

1.2.2 Protokoll zum Versenden von FEC-kodierten ADUs

Zusätzlich zu den eigentlichen FEC-kodierten Daten müssen bei dem Streamen von FEC-kodierten MP3 ADUs wichtige Metainformationen übermittelt werden, wie z.B. die Sequenznummer des Pakets innerhalb der versendeten Gruppe, die Sequenznummer der Paketgruppe, um bei umgeordneten Paketen und bei Paketverlusten ankommende Pakete in die richtige Gruppe zu ordnen, sowie ein Timestampfeld, um Aussetzer und Synchronisationsprobleme des Stroms zu erkennen und handzuhaben.

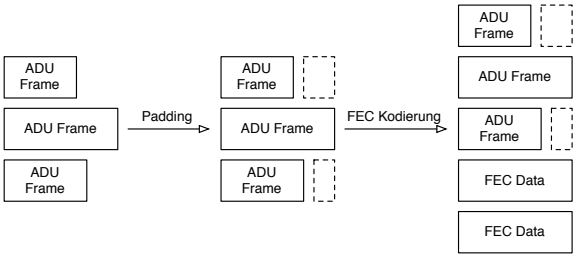

Abbildung 1.3: Verarbeitung von ADUs verschiedener Länge

Magic (0xFE)	Version (1)	Gruppensequ.	Paketsequ.
FEC k Parameter	FEC n Parameter	Maximale Länge der FEC Pakete	
Payload Länge		Timestamp (32 bits)	
Timestamp (32 bits)			

Abbildung 1.4: Format des FEC-Paket Headers

Auf dem Empfängerseite werden die FEC-kodierten Daten anhand der Sequenznummern in die richtigen FEC-Gruppen einsortiert. Ist beim Abspielen einer FEC-Gruppe diese Gruppe nicht komplett empfangen worden (d.h. es wurden weniger als k Pakete von n Paketen empfangen), werden nur die empfangenen Pakete mit Paketsequenznummer $< k$ ausgegeben (die wegen der systematischen Matrix gleich den Eingangs-ADUs sind), so dass nicht die komplette FEC-Gruppe verloren geht. Ist die Gruppe komplett empfangen worden, werden die Eingangs-ADUs mit dem Verfahren von Rizzo wiederhergestellt und zu MP3 Frames konvertiert, und anschliessend ausgegeben.