Technische Universität Ilmenau
Fakultät für Informatik und Automatisierung
Fachgebiet Telematik
Prof. Dr. Dietrich Reschke

Hauptseminar
Sommersemester 2004

# Fehlermodelle in Multimedia-Streams

Bearbeiter: Mario Weise
Betreuer: Dipl.-Inf. T. Strufe

# 1. Einleitung

## 1.1 Gegenstand der Arbeit

Im Zuge der immer schneller werdenden Anbindungen im privaten Bereich veränderte sich auch das Informationsangebot im Internet. Waren es früher noch reine textbasierte Seiten auf denen die Informationen präsentiert wurden, kamen im Laufe der Zeit mehr und mehr Bilder und andere multimediale Inhalte hinzu. Doch es dauerte bis zur Einführung von ISDN bis überhaupt an Multimedia-Streaming in Form von bewegten Bildern mit Ton zu denken war. Zu Beginn handelte es sich dabei nur um Videos mit geringer Qualität (niedrige Bitrate, Auflösung etc.). Diese wurden zusammen mit der zur Verfügung stehenden Bandbreite im Laufe der Zeit immer besser. Doch all der technische Fortschritt bis zum heutigen Tag brachte noch keine fehlerresistenten Verfahren zustande.
Deshalb ist es notwendig, sich als Anbieter einer Streaminglösung Gedanken um die Fehlertoleranz des Streams und die eventuell benötigten Fehlerkorrekturverfahren zu machen. Mit welchen Schwierigkeiten ist dabei zu rechnen?

- die diversen Datenverluste im Internet, welche seit der Einführung von drahtlosen Anbindungen sogar noch zunehmen
- die möglicherweise notwendige Erfordernis von Echtzeitübertragung des Streams (z.B. Livemitschnitte von Sportevents oder Konzerten)
- die nicht vorhersehbare Teilnehmerzahl und die damit schwankenden Übertragungslasten

Diese wenigen Punkte zeigen schon, dass ein nicht unerheblicher Aufwand auf die Anbieter zukommt.

## 1.2 Zielsetzung der Arbeit

Ziel dieser Arbeit soll sein, mögliche Fehlerarten in Multimedia-Streams zu untersuchen sowie deren Auswirkungen auf den Stream und die daraus resultierende Fehlerkorrektur zu erörtern.

## 1.3 Begriffsdefinitionen

Multimedia:

„Sammelbezeichnung für Produkte und Dienstleistungen aus dem Computer-, Telekommunikations-, Unterhaltungs- und Medienbereich. Grundlegende Merkmale von Multimedia-Anwendungen sind die gemeinsame Verwendung verschiedener statischer (Text, Foto und Grafik) und dynamischer (Audio, Animation und Video) Medientypen sowie insbesondere die Möglichkeit der interaktiven Nutzung. Interaktive Nutzung bedeutet, der Nutzer ist nicht nur ausschließlich Empfänger, sondern kann selbst über entsprechende Rückkanäle (Zwei-Wege-Technik) Inhalte abrufen und verändern bzw. Aktionen auslösen. Technologische Basis für Multimedia sind die digitale Technik, der Einsatz von Verfahren zur Datenkomprimierung, leistungsfähige Massenspeicher und Übertragungskanäle hoher Bandbreite (sog. Datenautobahnen) für vernetzte Anwendungen. […]“ [www.wissen.de]

Stream(ing):

„Bezeichnung für die kontinuierliche Datenübertragung von Audio- und Videodaten über das Internet mit dem Ziel einer Echtzeitübertragung. Am Endgerät erfolgt eine sofortige Decodierung. Es ist deshalb nicht notwendig, die Daten erst von einem Server abzurufen (Download), auf der eigenen Festplatte zwischenzuspeichern, um sie dann mit spezieller Software zu decodieren und abzuspielen. Streaming setzt keine Veränderungen im Übertragungsnetz voraus, aber für die Streaming-Technologie werden spezielle Plug-Ins im Browser benötigt (etwa RealAudio von RealNetworks).“ [www.wissen.de]

Codec:

„Als Codecs (englisches Akronym aus ***co****ding* und ***dec****oding*) bezeichnet man Verfahren, die aus einem Codierer und einem Decodierer bestehen.

Ein Codec kann in Hardware oder in Software realisiert sein. Es kann also auch in einem Programm enthalten sein, das Multimedia-Daten zwischen zwei Formaten hin- und herwandelt. Häufig werden von solchen Programmen die Daten dabei komprimiert.

Bei digitalen Videoformaten (MPEG, VP3, ...) und Audioformaten (MP3, AAC, OGG, ...) muss dasselbe Decodierverfahren verwendet werden, das auch zum Codieren verwendet wurde.“ [http://de.wikipedia.org/wiki/Codec]

## 2. Fehlerarten und –quellen

Dieses Kapitel soll die möglichen Fehlerquellen und die dort auftretenden Fehlerarten in Multimedia-Streams betrachten. Als Grundlage dient dazu ein einfaches Modell eines verteilten Systems. Dies deshalb, da es sich bei einer Sender-Empfänger-gestützten Streamingslösung um nichts anderes als ein verteiltes System handelt. Ein konzeptioneller Aufbau einer solchen Lösung kann wie folgt dargestellt werden.

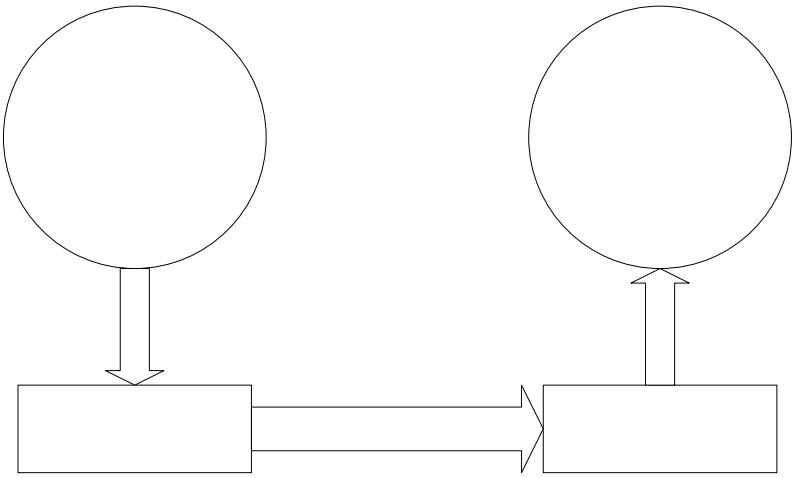


**Abbildung 2/1 Schema eines verteilten Systems**

Aufgrund dieses Schemas lassen sich die Fehlerquellen in zwei Kategorien unterteilen. Auf der einen Seite gibt es die so genannten Prozessfehler, welche sowohl auf der Sender- als auch der Empfängerseite auftreten können. Zum anderen lässt sich der Kommunikationskanal mit den Nachrichtenpuffern auf beiden Seiten als eine der Hauptfehlerquellen lokalisieren. Im weiteren Verlauf der Betrachtungen sollen die Fehlerarten für Prozess- und Kommunikationsfehler sofern möglich gesondert betrachtet werden. [laut [3]]

- **Ausfälle (Omission failures)**

Ausfallfehlern sind Fehler, bei denen ein Prozess oder Kommunikationskanal eine erwartete Aktion nicht ausführt. Der Absturz eines Prozesses ist dabei der Hauptausfallgrund. Das bedeutet, dass der Prozess gestoppt hat und keine weiteren Programmschritte mehr ausführen kann. Abhängig vom jeweiligen System können diese Fehler erkannt und notwendige Gegenmaßnahmen eingeleitet werden. Grundsätzlich dienen aber immer Zeitüberschreitungen (timeouts) zur Erkennung von Ausfallfehlern. Dabei wird in jedem Prozess ein Zeitintervall definiert, in welchem etwas „passieren“ muss. In synchronen Systemen können Prozesse

somit feststellen, ob andere Prozesse, mit denen sie kommunizieren, ausgefallen sind. Im Gegensatz dazu lässt sich dies in asynchronen Systemen nicht mit Sicherheit feststellen. Hier kann nicht unterschieden werden, ob ein Prozess abgestürzt ist, augenblicklich nur langsamer arbeitet oder die Antwort noch zum Empfängerprozess unterwegs ist.
Wie in Abbildung 2/1 zu sehen ist, wird bei der Kommunikation zwischen zwei Prozessen die betroffene Nachricht vom Sender über den Ausgangspuffer, den Kommunikationskanal und den Eingangspuffer zum Empfänger übertragen. Man spricht in diesem Modell von Ausfallfehlern auf dem Kommunikationskanal, sobald eine Nachricht auf selbigem verloren gegangen ist. Dies wird entweder durch einen Pufferüberlauf auf der Empfängerseite bzw. einem Vermittlungsknoten im Netz oder einem Übertragungsfehler verursacht. Letzteres lässt sich leicht durch in den Nachrichten integrierte Checksummen entdecken.

- **Zeitfehler (Timing failures)**

Wann immer Zeitschranken für Prozess- und Übertragungszeiten eingesetzt werden, können Zeitfehler in synchronen Systemen auftreten. Bei Prozessen kann es zum Beispiel passieren, dass die interne Uhr von der Realzeit im System abweicht und es somit zu Zeitüberschreitungen zwischen zwei Prozessen kommen kann, obwohl beide fehlerfrei ablaufen. Auch das Überschreiten eines Zeitintervalls zwischen zwei Schritten innerhalb eines Prozesses gehört zur Klasse der Zeitfehler. Im Kommunikationssystem spricht man von Zeitfehlern, wenn eine Nachricht vom Sender zum Empfänger länger braucht, als der Prozess beim Empfänger zur Bearbeitung vorgesehen hat.

- **Zufallsfehler (Arbitrary or Byzantine failures)**

Formatiert: Englisch (Großbritannien)

Hierbei handelt es sich um die schlimmstmögliche Fehlerart, da alle denkbaren Fehler zufällig auftreten können und unter Umständen nicht einmal als Fehler erkannt werden. Bei Prozessen kann dies bedeuten, dass der fehlerhafte Prozess falsche Programmschritte ausführt oder falsche Daten auf eine Anfrage sendet, ohne dass er selber oder die Empfängerprozesse vom Fehler Notiz nehmen. Im Gesamtsystem kann das wiederum zu einem unvorhersehbaren Verhalten führen. Kommunikationskanäle leiden unter Zufallsfehlern, wenn Daten manipuliert, nicht existente Nachrichten empfangen oder echte Nachrichten mehrmals empfangen werden. Solche Fehler treten aber selten auf, da die Kommunikationssoftware diverse Mechanismen (z.B. Checksummenbildung) zur Fehlererkennung zur Verfügung stellt.

- **Verschleierungsfehler (Masking failures)**

Verschleierungsfehler sind genau genommen keine Fehler im eigentlichen Sinne. Vielmehr wird versucht, eine Fehlerart so zu maskieren, dass sie nach außen nicht sichtbar ist. Ein Beispiel hierfür wäre eine Multi-Server-Architektur, bei der ein Server den Ausfall eines anderen kompensiert, indem er dessen Dienste bzw. Prozesse übernimmt und das System somit fehlerfrei nach außen weiter arbeitet.

# 3. Fehlerbehebung

Das nun folgende Kapitel wird sich mit der Behebung von Fehlern in Multimedia-Streams befassen. Dabei soll zunächst der prinzipielle Ablauf einer Fehlerkorrektur gezeigt werden, um dann im weiteren Verlauf auf spezielle Lösungsansätze einzugehen [laut [1]].
Der prinzipielle Aufbau einer Sender-Empfänger-Kodierungsstruktur lässt sich wie folgt darstellen:

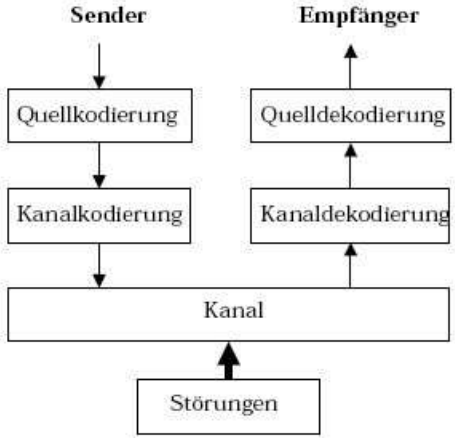



**Abbildung 3/1 Funktionale Gliederung eines Streamingsystems**

Abhängig von der Funktionsweise der Korrekturverfahren sind diese dann entweder im Quell- oder Kanal(de)kodierer des Senders oder Empfängers angesiedelt. Grund hierfür ist die Tatsache, dass jedem Teil des Gesamtsystems andere Informationen über den Stream zur Verfügung stehen und somit verschiedene Maßnahmen gefällt werden können.
Als nächstes soll eine konzeptionelle Aufteilung der Fehlerbehebung unter Berücksichtigung der Sender-Empfänger-Struktur vorgenommen werden.

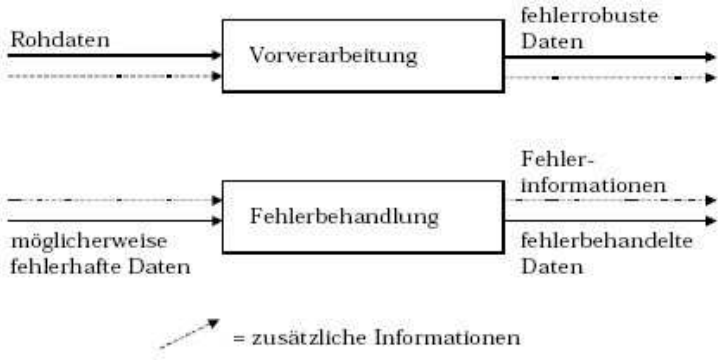



**Abbildung 3/2 Schema einer verteilten Fehlerbehebung**

Abbildung 3/2 zeigt die Bearbeitung der Rohdaten im Sender dahingehend, dass diese durch Hinzufügen von Daten fehlerrobuster an den Empfänger geschickt werden können. Mit Hilfe der enthaltenen Zusatzinformationen im Stream ist es dem Empfänger möglich, eventuell fehlerhafte Daten (bis zu einem gewissen Grad) wiederherzustellen.

## 3.1 Senderbasierte Techniken

Der folgende Abschnitt befasst sich mit den senderbasierten Fehlerbehandlungstechniken. Diese lassen sich anhand des in Abbildung 3.1/1 gezeigten Schemas in aktive und passive Techniken einteilen. Aktiv bedeutet in diesem Zusammenhang, dass der Sender nur aufgrund einer Nachricht des Empfängers reagiert. Unter passiven Techniken versteht man demzufolge das Gegenteil, das heißt, der Sender ergreift selbstständig ohne Zutun des Empfängers die notwendigen Maßnahmen.

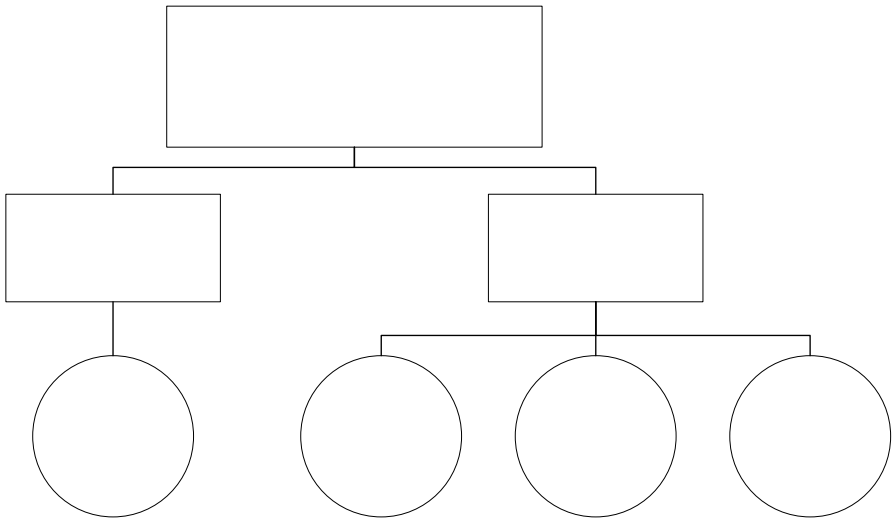


**Abbildung 3.1/1 Übersicht der senderbasierten Techniken**

### 3.1.1 Wiederholung auf Anfrage (Automatic Repeat on Request - ARQ)

Beim ARQ handelt es sich um den einfachsten aller denkbaren Ansätze. Der Empfänger kann mittels einer Nachricht an den Sender ein verloren gegangenes Paket neu anfordern. Man spricht in diesem Fall von aktiver Fehlerkorrektur, weil der Sender auf eine Anfrage des Empfängers wartet und nicht aus eigenen Überlegungen ein Paket neu versendet.

senderbasiert



aktiv

p

Für diesen Ansatz müssen jedoch folgende Problemstellungen berücksichtigt werden:

- Kann das neu angeforderte Paket überhaupt innerhalb des Verarbeitungszeitraums beim Empfänger eintreffen?
- Kann die Wiederholung von Informationen ein besseres Ergebnis als die empfängerseitige Fehlerbehebung liefern?

Um diese Probleme zu lösen, ist es nötig, dass entsprechende Modelle (Zeit- bzw. Fehlertoleranzmodelle) in der Anwendung implementiert werden, welche die Notwendigkeit einer Neuanforderung eines Paketes abschätzen können. Schon diese beiden Probleme zeigen aber, dass diese Technik nur bedingt für Multimedia-Streams geeignet ist. Denn durch das wiederholte Senden der Pakete entsteht eine nicht zu unterschätzende Last auf dem Kommunikationskanal. Des Weiteren muss der Sender die Anfragen sowohl bearbeiten als auch gesendete Pakete zwischenspeichern können. Aber genau diese zusätzliche Last auf dem Gesamtsystem führt gerade bei großen Teilnehmerzahlen dazu, dass ein entgegen gesetzter Effekt, nämlich noch mehr Paketverluste, erzeugt wird.

### 3.1.2 Datenstreuung (Interleaving)

Normalerweise liegt es nahe, mehrere Dateneinheiten des Quellkodierers (z.B. GOPs, Frames usw.) in einem Datenpaket zusammenzufassen. Geht dieses jedoch verloren, entsteht eine große Lücke im Stream. Bedenkt man nun, dass zum einen der Mensch kleine Bild- und/oder Tonausfälle schlechter wahrnimmt als große und empfängerbasierte Fehlerkorrekturtechniken besser bei kleinen und weit verstreuten als bei großen Lücken arbeiten (siehe Kap. 4.4), macht es Sinn, zusammenhängende Daten nicht in einem Paket zu übermitteln. Abbildung 3.1/2 zeigt ein Beispiel mit 4 Dateneinheiten pro Paket. Was diese Dateneinheiten im Einzelnen enthalten, spielt in diesem Fall kein Rolle.

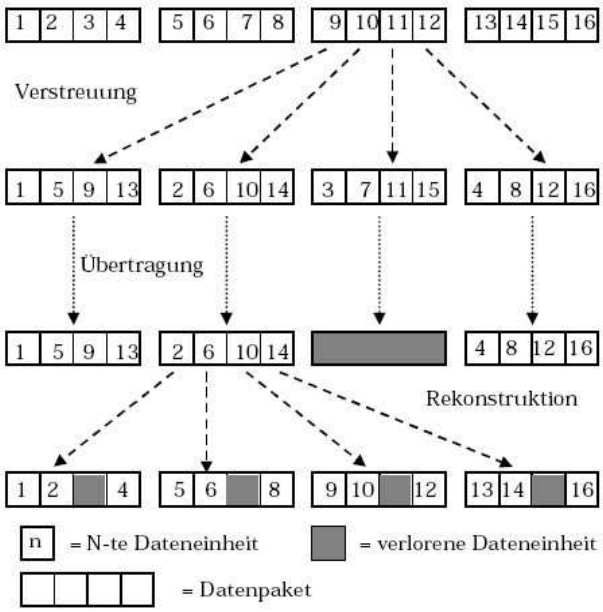



**Abbildung 3.1/2 Interleaving**

Eine solche Technik müsste im Quellkodierer des Senders untergebracht werden, denn nur dieser hat genügend Informationen über den Aufbau des Streams. Dies zieht natürlich nach sich, dass im Dekodierer beim Empfänger die gleiche Technik zur korrekten Wiederherstellung der Daten implementiert sein muss. Gegen eine Ansiedlung im Kanalkodierer spricht vor allem die Tatsache, dass dieser keinerlei Information über den Inhalt der Datenpakete hat und somit ohne zusätzliche Informationen die Daten auch nicht im Sinne einer guten Fehlertoleranz streuen kann.

### 3.1.3 Datenstreuung (Information Dispersal)

Ein weiterer Ansatz zur Datenstreuung ist die Übermittlung der Datenpakete über disjunkte Pfade zu den Empfängern. Dabei wählt der Sender automatisch verschiedene Einwahlknoten in das Netz. Vorteil dieses Ansatzes ist, dass bei einem Ausfall einer Datenleitung (z.B. Pufferüberlauf an einem Vermittlungsknoten) nicht mehrere zusammenhängende Datenpakete verloren gehen, dass die Last gleichmäßiger über dem gesamten Kommunikationsnetz verteilt und auch keine höhere Last durch Zusatzinformationen im Stream verursacht wird.
Doch wie jeder Ansatz, bringt auch dieser einige Nachteile mit sich. Allen voran die vom Sender nicht berechenbaren Laufzeiten der Datenpakete über die verschiedenen Pfade. Somit kann nicht gewährleistet werden, dass die Pakete zum notwendigen Verarbeitungszeitpunkt beim Empfänger eintreffen. Eine mögliche Lösung hierfür wäre ein im Sender implementiertes Zeitmodell, welches die Laufzeiten auf den Pfaden misst und für die Zukunft

abschätzen kann. Ein weiterer Nachteil ist, dass der Sender keinen direkten Einfluss auf den Verlauf der Pfade hat. Das heißt, es kann durchaus passieren, dass sich ein oder mehrere Pfade irgendwann einmal treffen und parallel weiterlaufen. Dass dadurch der Grundgedanke des Ansatzes zerstört wird, liegt auf der Hand.
Eine mögliche Implementierung dieser Technik müsste im Kanalkodierer erfolgen und könnte so aussehen, dass erst bei Bedarf (z.B. Überlastung der Primärverbindung) eine weitere Verbindung aufgebaut wird.

### 3.1.4 Skalierung

Der Kernpunkt dieser Technik besteht darin, die Kodierung des Streams zur Laufzeit so anzupassen, dass auf die aktuelle Situation (z.B. Fehlerrate, Netzlast) umgehend reagiert werden kann. So wäre es denkbar, bei blockbasierten Kodierungen den Quantisierungsfaktor, die Bildwiederholrate oder gar die komplette Blockzusammenstellung zu ändern.
In der Praxis könnte das ganze dann so aussehen, dass mittels Sitzungs- und Netzmanagement dem Quellkodierer vorgeschrieben wird, wie der Stream zu kodieren ist. Bei Netzüberlastung wird die Datenrate reduziert oder bei vermehrten Paketverlusten die Anzahl der zueinander unabhängigen Bilder (I-Frames) erhöht, um eine Fehlerfortpflanzung auf Empfängerseite zu vermeiden. Eine mögliche Anpassung der Quellkodierung bei zunehmender Fehlerrate ist in Abbildung 3.2/3 zu sehen.

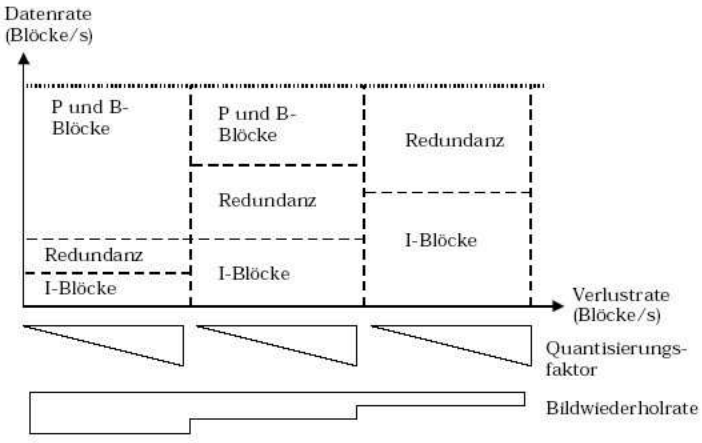



**Abbildung 3.1/3 Skalierung**

### 3.1.5 Vorwärtsfehlerkorrektur (bitorientiert)

Bei der Vorwärtsfehlerkorrektur wird versucht, mittels Redundanz in den Datenpaketen verlorene Pakete bzw. Blöcke wieder komplett herzustellen. Dabei wird das i-te Bit eines Datenpaketes zum i-ten Bit des Redundanzpaketes, welches mit übertragen wird. Die folgende Abbildung zeigt ein Beispiel mit 3 Datenpaketen.

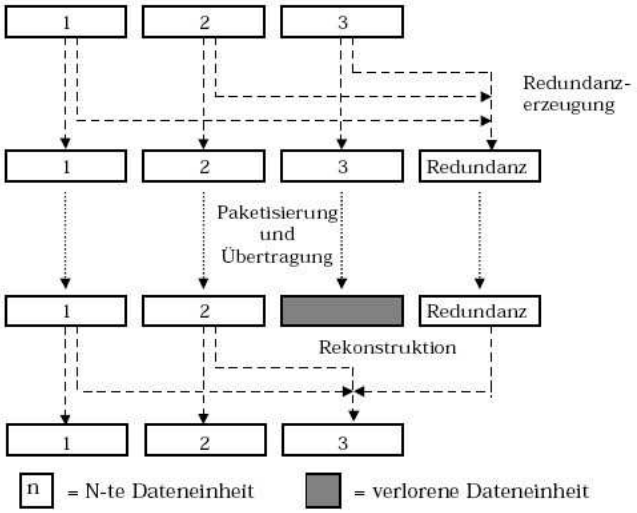



**Abbildung 3.1/4 Bitorientierte Vorwärtsfehlerkorrektur**

Es gibt diverse Möglichkeiten, die Redundanzdatenpakete zu erzeugen. Als Beispiel sei zum einen die Paritätskodierung genannt, bei der mit Hilfe einer logischen XOR-Funktion über eine Anzahl von Datenpaketen so genannte Paritätspakete erzeugt werden, mit dessen Hilfe dann verlorene Pakete wieder hergestellt werden können. Beim Reed-Solomon-Kode dagegen wird die Redundanz mit Hilfe eines Polynoms und dessen algebraischen Eigenschaften erzeugt. Der Reed-Solomon-Kode gehört zur Klasse der linearen, zyklischen Kodes. Die genaue Aufschlüsselung dieses Verfahrens würde den Rahmen dieser Arbeit aber bei weitem sprengen. Deshalb sei an dieser Stelle auf [5] verwiesen.

Die Vorteile der bitorientierten Vorwärtsfehlerkorrektur liegen auf der Hand. Sie ist unabhängig vom Inhalt der Datenpakete, weswegen sie im Kanalkodierer implementiert werden kann. Außerdem können die Redundanzpakete ohne großen Rechenaufwand erzeugt werden und alle erkannten Fehler lassen sich hundertprozentig wiederherstellen. Als Nachteile sind die erhöhte Netzlast durch die Redundanz sowie eventuelle Verzögerungen durch den Dekodierer bei der Datenwiederherstellung zu nennen. Außerdem kann die

Vorwärtsfehlerkorrektur keinen stufenlosen Qualitätsabfall bieten, das heißt, dass zum Beispiel ein Bild entweder optimal wiederhergestellt oder komplett verloren ist.

### 3.1.6 Vorwärtsfehlerkorrektur (inhaltsorientiert)

Im Gegensatz zur bitorientierten Vorwärtsfehlerkorrektur ist die inhaltsorientierte Vorwärtsfehlerkorrektur im Quellkodierer implementiert, da dieser, wie der Name schon sagt, das Wissen über den Aufbau der Quelldaten nutzt. Der Grundgedanke hier ist es, mit Hilfe von zwei (unter Umständen auch mehreren) qualitativ unterschiedlichen Kodierungen der Quelldaten Redundanz zu erzeugen und diese über den Kommunikationskanal zum Empfänger zu schicken. Bei einer Bildübertragung bedeutet das, dass die Sekundärkodierung stärker quantisiert ist, oder eine geringere Bildwiederholfrequenz hat (siehe auch 3.1.4) und im Falle eines Übertragungsfehlers in der Primärkodierung diese ersetzt. Für den Empfänger äußert sich dieses Umschalten nur in einer sichtbaren Qualitätsabnahme des Stream, nicht aber in einem kurzzeitigen Totalausfall. Ein Beispielablauf dieses Verfahrens ist in Abbildung 3.1/5 zu sehen.

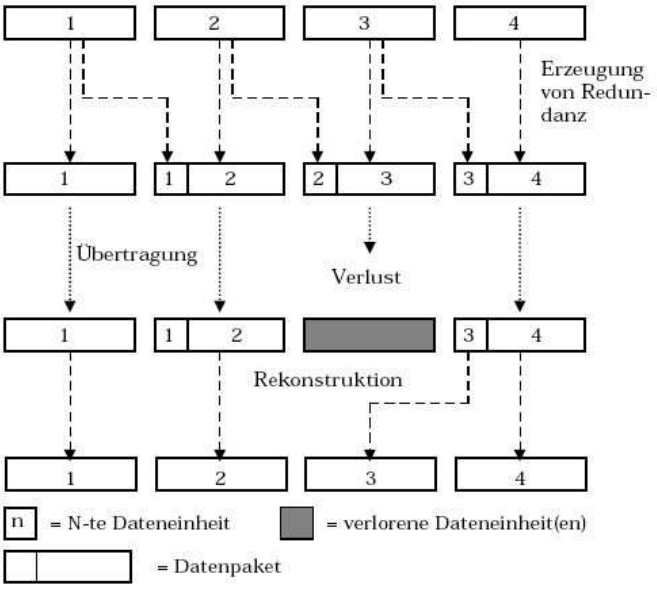
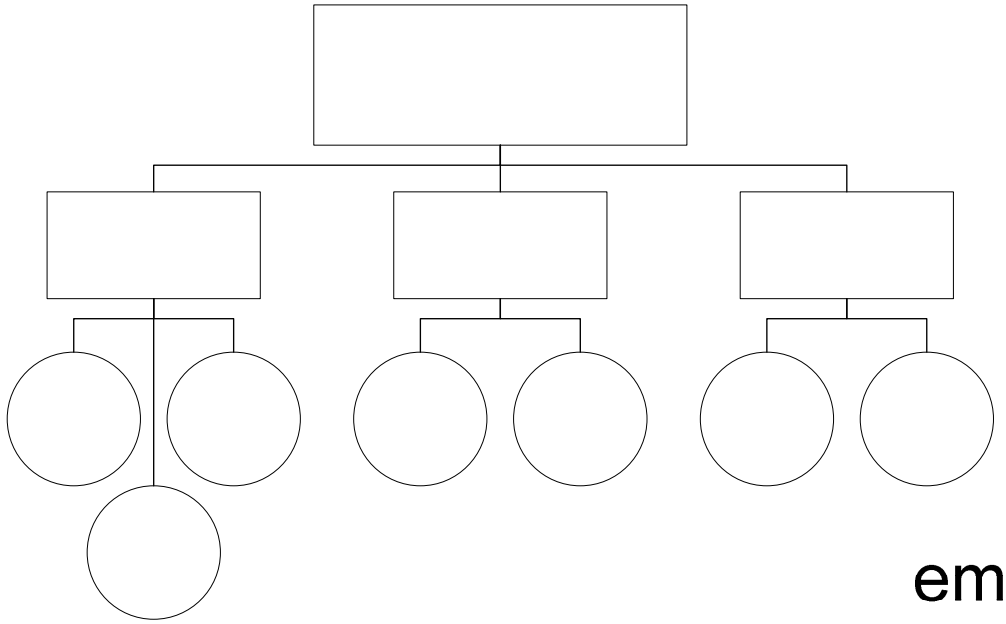



**Abbildung 3.1/5 Inhaltsorientierte Vorwärtsfehlerkorrektur**

## 3.2 Empfängerbasierte Techniken

Nachdem in Kapitel 3.1 die senderseitige Vorverarbeitung der Daten behandelt wurde, soll dieses Kapitel nun die empfängerseitige Fehlerbehandlung betrachten. Die folgende Übersicht zeigt die zu betrachtenden Techniken im Einzelnen.



empfängerbasier

**Abbildung 3.2/1 Übersicht der empfängerbasierten Techniken**

### 3.2.1 Spleißen (Splicing)

Beim Spleißen handelt es sich um die einfachste der drei Techniken des so genannten Einfachen Einfügens. Ohne Wissen über den Inhalt der Daten wird eine Lücke im Stream mit Nulleinheiten gefüllt (siehe Abb. 3.2/2).

Einfaches

Geglättetes Einfügen

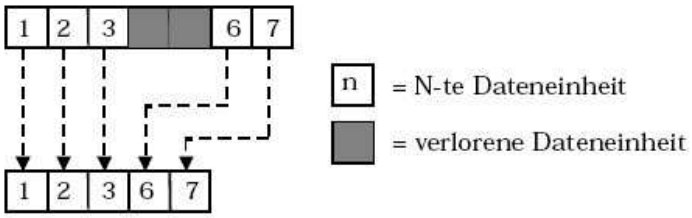



**Abbildung 3.2/2 Spleißen**

Wiederholung

Spleißen

vorheriger Daten

Wellenmuster suchen

Zeitstrecku

Aber gerade wegen der Einfachheit dieser Technik bringt sie viele Nachteile mit sich. So wird das zeitliche Verhalten des Stream verändert, was vor allem bei Audiodaten schon ab ein



Einfügung von Stille

paar Millisekunden nicht mehr tolerierbar ist. Handelt es sich bei den verloren gegangenen Dateneinheiten um ganze Bilder eines Videostreams, äußert sich Spleißen in Form eines Abfalls der Bildwiederholrate, was spätestens ab eine Rate von unter 15Bilder/sec. nicht mehr tragbar ist.

### 3.2.2 Einfügen von Stille

Im Gegensatz zum Spleißen bleibt bei diesem Verfahren das zeitliche Verhalten des Streams erhalten. So wird hier die entstandene Lücke nicht mit Nulleinheiten, sondern mit zufällig erzeugten Dateneinheiten gefüllt (siehe Abb. 3.2/3). Im Falle von Audiodaten können diese Fülleinheiten z.B. Hintergrundrauschen enthalten, welches vom Menschen als weniger störend empfunden wird als reine Tonaussetzer. Bei Bilddaten hängen die erzielten Ergebnisse stark von der Art des Verlustes ab. Ist ein komplettes Bild oder ein größerer Bildteil verloren gegangen, kann man versuchen, einfarbige Fülleinheiten zu verwenden. Besser ist es jedoch, auf psychovisuell unauffällige Muster zurückzugreifen. Anders sieht es aus, wenn Teile eines Makroblockes innerhalb eines Bildes verloren sind. Fehlen zum Beispiel die Luminanzinformationen, wären auch die vollständigen Chrominanzblöcke nutzlos. Deshalb setzt man die Luminanzwerte entweder auf Null oder auf statistisch ermittelte Standardwerte, welche als weniger störend empfunden werden, um die vorhandenen Daten nutzbar zu machen. Aber wie beim Spleißen gilt auch hier, dass die Ergebnisse mit größer werdenden Lücken stark in der Qualität nachlassen.

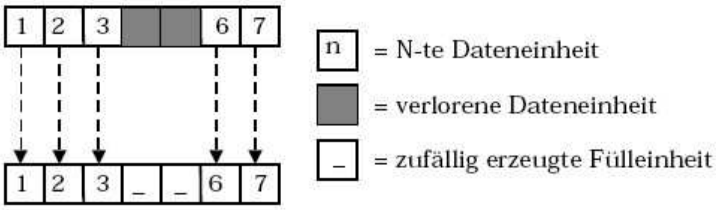



**Abbildung 3.2/3 Einfügen von Stille**

### 3.2.3 Wiederholung vorheriger Daten

Nachdem die ersten beiden Verfahren (siehe 3.2.1 und 3.2.2) keinerlei Informationen aus den zuvor korrekt übertragenen Daten zur Lückenfüllung zur Hilfe nahmen, geschieht genau das bei dieser Technik. Dabei kann es sich um eine einfache sowie eine sich wiederholende Wiederholung handeln. Letztere macht bei Audiodaten in Kombination mit einer Abschwächung jeder Wiederholung vor allem bei größeren Lücken Sinn, da das menschliche Hörempfinden diese Art besser aufnimmt als eine lange Wiederholung eines großen Bereichs.

Anwendung findet dieses Verfahren zum Beispiel in GSM-Netzen. Trotzdem gilt auch hier, je größer die Lücke ist, desto größer ist auch der Qualitätsabfall. Ein Beispiel für eine Wiederholung von Daten ist in Abbildung 3.2/4 zu sehen.

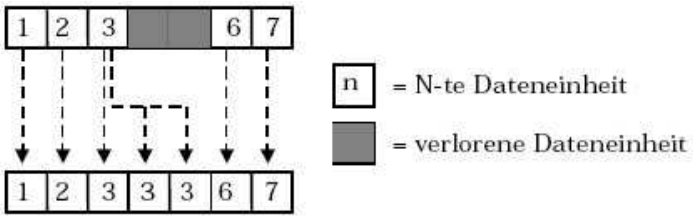



**Abbildung 3.2/4 Wiederholung vorheriger Daten**

Im Falle von Bilddaten kommt es wieder darauf an, ob ein ganzes Bild bzw. mehrere Bilder oder nur Bildteile verloren sind. Bei kompletten Bildausfällen äußert sich dieses Verfahren in Form von so genannten „frozen frames“. Das Bild bleibt für die Dauer der Lücke stehen und läuft danach normal weiter. Nachteil dieser Technik ist der sprunghafte Szenenwechsel beim Wiedereinsetzen der korrekten Daten. Handelt es sich bei den Verlusten nur um Bildteile, zum Beispiel Makroblöcke, bleibt nicht das ganze Bild sondern nur genau der verlorene Block stehen. Dies fällt mitunter nicht auf, wenn es sich um eine eher statische Szene handelt und der Ausfall irgendwo im Bildhintergrund stattfindet.

### 3.2.4 Zeitstreckung

Bei der Zeitstreckung handelt es sich um eine einfache Fehlerkorrekturmethode. Wie in Abbildung 3.2/5 zu sehen, werden die Dateneinheiten um die Lücke soweit gestreckt, bis diese gefüllt ist.

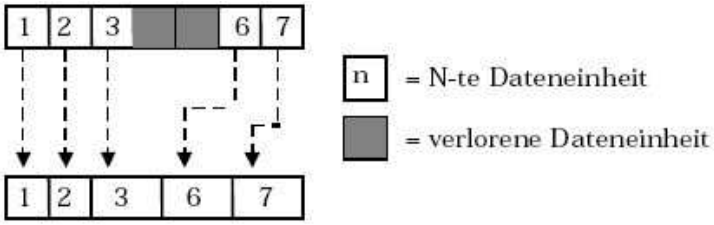



**Abbildung 3.2/5 Zeitstreckung**

Besonders bei Audiodaten ist dieses Verfahren sehr aufwendig, denn es muss versucht werden, die Kontinuität des Streams zu erhalten. Dazu müssen die Tonhöhen, die Amplituden

etc. angepasst werden. Dies ist aber immer noch mit einem sehr hohen Rechenaufwand verbunden.
Deswegen bietet sich diese Prozedur eher für die Korrektur von Bilddaten an. Bei einer Bildwiederholrate von 25 Bildern/Sec. ist ein Bild genau 40ms zu sehen. Fehlt nun ein Bild, ist sowohl das Bild davor als auch danach für 60ms zu sehen. Dies fällt aber dank der Trägheit des menschlichen Auges bei solchen Größenordnungen kaum auf.

### 3.2.5 Wellenmustersuchen

Ein Nachteil aller bisher genannten empfängerseitigen Fehlerkorrekturverfahren ist, dass die Fülldaten mehr oder weniger zufällig ausgewählt werden und somit dementsprechende Ergebnisse geliefert werden. Die Wellenmustersuche beruht darauf, dass die korrekt übermittelten Daten vor und nach der Lücke zur statistischen Analyse herangezogen und daraus die Fülleinheiten erzeugt werden (siehe Abbildung 3.2/6).

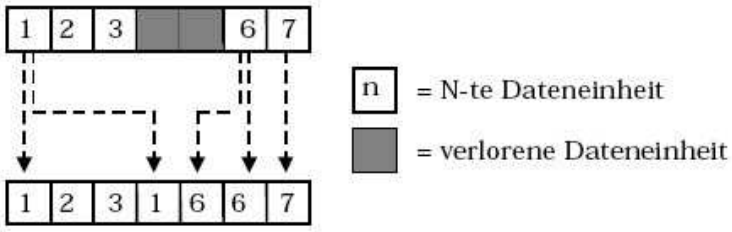



**Abbildung 3.2/6 Wellenmustersuchen**

Eine Variante der Wellenmustersuche bei der Behandlung von Audiodaten ist die schablonierte Wellenmustersuche. Dabei wird mit Hilfe von vordefinierten Schablonen entweder eine ein- oder zweiseitige Datenanalyse an der Lücke vorgenommen. Ziel ist es, dem Hörer eine möglichst große Kontinuität im Stream als Ergebnis zu präsentieren. Bei der einseitigen Suche wird das gefundene Muster wie im vorangegangenen Kapitel in der Lücke wiederholt. Dagegen wird bei der zweiseitigen Analyse mit Hilfe von interpolierten Daten aus vor und nach der Lücke gefundenen Mustern gearbeitet. Dass diese Methode die besseren Ergebnisse liefert, liegt auf der Hand. Eine weitere Art der Wellenmustersuche basiert auf Algorithmen anstatt fixer Schablonen. Man spricht in diesem Fall auch von flexiblen Schablonen. Diese Methode ist mitunter Ressourcen schonender, da anhand der Lückengröße entschieden wird, welche Algorithmen angebracht sind, und liefert eine leicht bessere Korrektur als das Schablonenverfahren.

Die einfachste Möglichkeit mit dieser Technik Bilddaten wieder herzustellen ist eine Mittelwertbildung der angrenzenden korrekten Bildbereiche. Für den Beobachter äußert sich dieser Effekt mit einer lokalen Unschärfe im korrigierten Bereich. Die Verwendung von diversen Interpolationsalgorithmen stellt ähnlich wie bei der Audiobehandlung auch hier eine Verbesserung der Endergebnisse dar. Vor allem die Bildung von nicht vorhandenen Kanten an den Übergängen zwischen der Lücke und dem Bild kann damit vermindert werden. Eine weitere Entwicklungsstufe wäre die Verwendung von MPEG4-basierten Videocodecs, welche anstatt einer blockbasierten eine objektbasierte Quellkodierung vornehmen. Somit ist es möglich, das Bild nach Objekten zu durchsuchen, die Kanten dieser in die Lücke zu verlängern und somit eine noch „weichere“ Korrektur vorzunehmen.

### 3.2.6 Rückgriff auf Zustandsinformationen

Grundlage dieses Verfahrens ist die Verwendung von codec-spezifischen Informationen innerhalb des Streams. Somit ist das Verfahren vom Codec abhängig. Das Spektrum reicht von Interpolation bis hin zum Rekodieren des verlorenen Stückes. In dieser Arbeit soll auf entsprechende Codecs und deren Funktionsweise nicht eingegangen werden, deshalb sei dazu auf andere Arbeiten verwiesen. Aus diesem Grund scheinen die Datenpakete 4* und 5* in Abbildung 3.2/7 ohne Bezug zu den anderen zu entstehen.

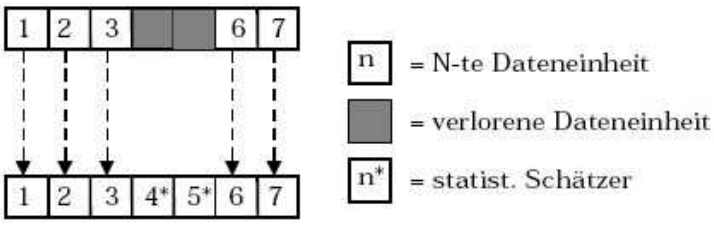



**Abbildung 3.2/7 Rückgriff auf Zustandsinformationen**

### 3.2.7 Modellbasierte Schätzung

Der grundlegende Ansatz dieser Prozedur ist ähnlich der in Kapitel 3.2.5 gezeigten Wellenmustersuche, geht dabei aber einige Schritte weiter. Im Falle von Audiodaten werden diese auf beiden Seiten der Lücke an ein statistisches Modell angepasst, welches dann benutzt wird, die Lücke mit den künstlich erzeugten Daten zu füllen (siehe Abbildung 3.2/8).

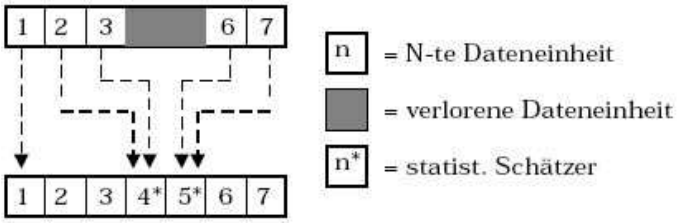



**Abbildung 3.2/8 Modellbasierte Schätzung**

Die Schwierigkeit liegt darin, überhaupt ein solches Modell aufzustellen. Denn je größer dieses wird, desto größer wird auch der Rechenaufwand bei der Anpassung, was wiederum ab einem gewissen Grad die Anwendung dieses Verfahrens nutzlos macht. Bei Videodaten wird es dann noch komplexer, denn diese besitzen mehr Dimensionen als ein linearer Audiostream. Somit wird das Modell automatisch größer, was auch hier den Rechenaufwand explizit ansteigen lässt und eine Anwendung vor allem unter Realzeitanforderungen zurzeit noch in Frage stellt. Zukünftig stellt es aber sicher eine Alternative dar, denn die Ergebnisse sind allen anderen Verfahren voraus.

# 4. Zusammenfassung und Ausblick

Die Vielzahl der in dieser Arbeit vorgestellten Techniken lässt erahnen, wie schwer eine geeignete Auswahl für eine Implementierung fallen kann. Abhängig von der Art der vorkommenden, aber gleichzeitig kaum vorhersehbaren Fehler ist ein passendes Verfahren zu wählen. Eine optimale Lösung scheint es dabei nicht zu geben. Nichtsdestotrotz haben sich einige Techniken als „alltagstauglicher“ als andere erwiesen. So wird beispielsweise das Interleaving von den meisten Streamingformaten eingesetzt, da es die empfängerseitigen Korrekturmechanismen erheblich unterstützt (siehe 3.1.2).

So muss das Ziel eines Entwicklers sein, eine Fehlerevaluierungsfunktion aufzustellen, welche anhand von statistischen Prognosen, Modellen und Echtzeitanalysen eine geeignete Fehlerkorrektur auswählt, um dem Betrachter zu jeder Zeit die bestmögliche Qualität präsentieren zu können.

# Abbildungsverzeichnis



# Quellenverzeichnis

[1] Suchanek, T. (2000), „Untersuchung von Fehlertoleranztechniken zur realzeitorientierten Videokommunikation“

[2] Hoffmann, M. (2003), “On Failure Semantics for Multimedia Streaming Applications”

[3] Coulouris, G. / Dollimore, J. / Kindberg, T. (2001, 3rd Edition), “Distributed Systems – Concepts and Design”

[4] Tanenbaum, A. (1998, 3. revidierte Auflage), „Computernetzwerke“

[5] Sylvester, J. (2001), „Reed-Solomon-Code”, url: http://www.csupomona.edu/~jskang/files/rs1.pdf

[6] Prof. Dr.-Ing. Schade, H.P. (2003), Vorlesungen zur digitalen Audio- und Videotechnik url: http://www.imt.tu-ilmenau.de/lehre