Hochschule Konstanz Technik, Wirtschaft und Gestaltung (HTWG)
Fakultät Informatik
Rechner- und Kommunikationsnetze
Prof. Dr. Dirk Staehle

Labor zur Vorlesung Kommunikationstechnik

Convolutional Code (Python)

Prof. Dr. Dirk Staehle

Die Abgabe erfolgt durch Hochladen der bearbeiteten Word-Datei in Moodle.

Bearbeitung in Zweier-Teams

Team-Mitglied 1:

Team-Mitglied 2:

1 Einleitung

In dieser Aufgabe soll die Übertragung eines Frames mit einem Convolutional Code zur Forward Error Correction (Fehlerkorrektur) simuliert werden. Das Ziel der Aufgabe ist, den besten Convolutional Code in Abhängigkeit der Bitfehlerrate zu bestimmen.

2 Convolutional Code

Wir betrachten in dieser Aufgabe den WLAN Faltungs-Code in Abbildung 1 mit eine nativen Code-Rate von 1/2 sowie die in Abbildung 2 und Abbildung 3 gegebenen Punktierungsmuster für Code-Raten von 3/4 und 2/3. Zusätzlich betrachten wir eine Convolutional Code mit Rate 1/3, der in oktaler Notation als 155,173,157 gegeben ist.

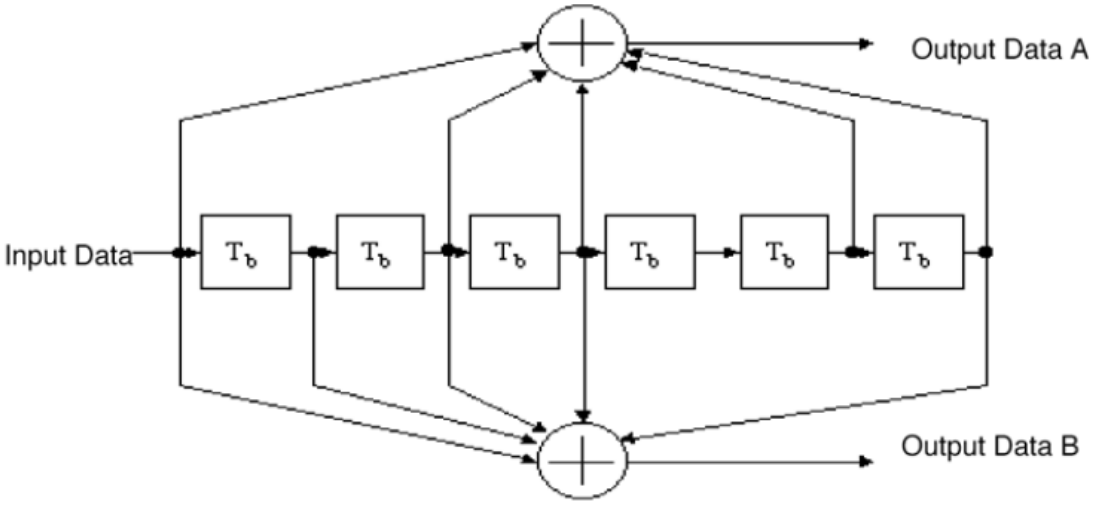

Abbildung 1 Convolutional Encoder

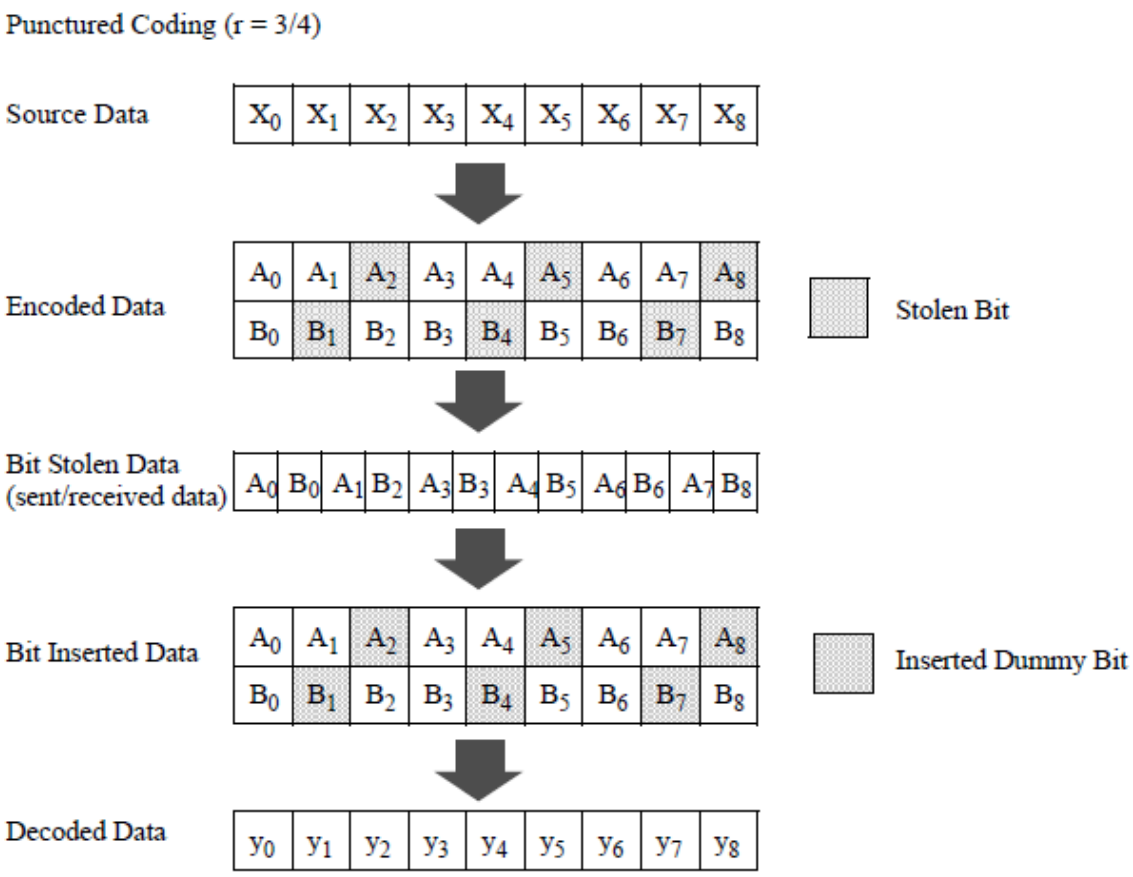

Abbildung 2 Punktierung für Coderate 3/4

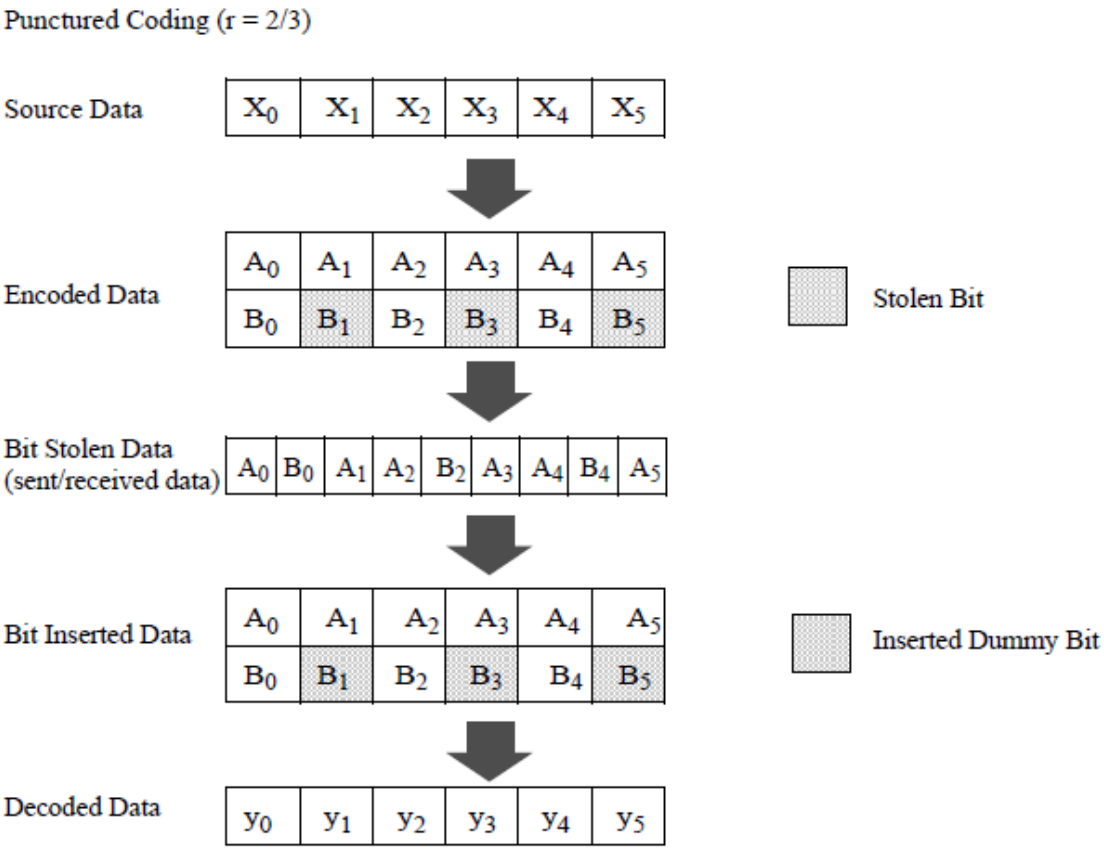

Abbildung 3 Punktierung für Coderate 2/3

3 Simulation

Implementieren Sie eine Funktion, die die Übertragung mehrere Nachrichten simuliert und statistisch auswertet. Die Funktion soll zufällige Nachrichten erzeugen, die Nachricht mit dem Convolutional Code kodieren, zufällige Bitfehler erzeugen, die Fehlerkorrektur durchführen und die resultieren Framefehlerrate bestimmen. Implementieren Sie konkret die Funktion `simulate_cc`(code,k,M,pb,s) mit den folgenden Schritten:

1. Zufallszahlengenerator mit Seed `s` initialisieren
2. M zufällige Nachrichten (Frames) mit einer Länge von k Bits erzeugen
3. Alle Nachrichten mit dem Convolutional Code `code` codieren
4. Bitfehler mit der Bitfehlerwahrscheinlichkeit pb erzeguen
5. Für alle Nachrichten die Fehlerkorrektur durchführen
6. Die resultierende Framefehlerrate bestimmen und zurückliefern

Tipp:

Verwenden Sie die Klassen `ConvolutionalCode` und `TerminatedConvolutionalCode` mit dem Mode „zero-termination“ aus dem Modul `komm`. Der „`zero-termination`“ Mode hängt die benötigten Nullen bei der Codierung an und entfernt sie bei der Dekodierung wieder. Beachten Sie, dass die Klasse ConvolutionalCode im Konstruktor die oktale Notation direkt unterstützt, d.h. einen Faltungscode mit oktaler Notation 753,533 würden Sie mit `ConvolutionalCode([[0o753, 0o533]])` erzeugen.

Beachten Sie weiterhin, dass die Codierung nicht exakt wie in der Vorlesung erfolgt. Das Input-Bit wird rechts eingefügt und mit dem niedrig-wertigsten Bit des Feedback-Polynoms verknüpft. Sie können das in der Aufgabe vernachlässigen.

Das Modul `komm` unterstützt keine Punktierung, so dass Sie diese selbst implementieren müssen. Verwenden Sie dazu bei der Dekodierung mit der Methode `decode()` die Methode 'viterbi_soft' mit der folgenden Codierung für ihre empfangene Nachricht:

Empfagenes Bit	Codierung
0	1
1	-1
punktiert	0

4 Simulationsstudie

Vergleichen Sie in einer Simulationsstudie die Framefehlerrate (FER) für die Code-Raten 1/3, 1/2, 2/3 und 3/4. Verwenden Sie eine Framegröße von 64 Bit und Bitfehlerwahrscheinlichkeiten von 0% bis 20%. Bestimmen Sie außerdem den erzielten normierten Durchsatz, d.h. die pro übertragenem Bits de facto übertragenen Nutzbits also: Durchsatz = Code-Rate x (1-FER)

Stellen Sie in zwei Grafiken einmal den Zusammenhang zwischen der Bitfehlerwahrscheinlichkeit p und der Framefehlerrate und einmal den Zusammenhang zwischen der Bitfehlerwahrscheinlichkeit p und dem Durchsatz dar. Vergleichen Sie in den Grafiken jeweils die 4 Coderaten.