9. Übung zur VL Betriebs- und Kommunikationssysteme

Tutor: Thomas Tegethoff

Bearbeiter: Etienne Jentzsch, Carola Bothe

1a) Data-Link Layer

Der Data-Link Layer ist die Schicht zwischen dem Physical Layer, welches Bits in physikalisches Signal umwandelt und dem Network-Layer, welches Datenpakete erwartet. Sie steuert zunächst den Datenfluss, bestimmt also über die MAC (medium access control) Adresse, wohin gesendet wird und wann gesendet werden darf. Damit der Empfänger nicht überflutet wird und erkannt wird, wenn Pakete verloren gehen, wird das stop-and-wait Prinzip genutzt: Der Sender wartet auf ein 'okay, ich habe alles richtig empfangen'-signal vom Empfänger bevor er erneut senden kann. Eine weitere Aufgabe ist das Aufteilen in frames mit flags. Dabei wird um die Nutzlast ein Rahmen bestehend aus Header und Trailer gebaut, der z.B. die Länge des Signals oder andere Metadaten enthält und/oder mit flags umrahmt, die zeigen, wann die Nutzlast beginnt und endet. Damit die flags eindeutig bleiben, wird bitstuffing genutzt, also bei der Nutzlast nach fünf Einsen eine Null eingeführt. Die letzte Funktion besteht in dem Bilden einer Checksumme, zur Überprüfung der Übertragung. Mit CRC weiß man dabei mit einer Sicherheit von ca. 98%, dass wenn die Prüfsumme stimmt, die Übertragung fehlerfrei geklappt hat. Im Data-Link Layer wird diese also berechnet und an die Nutzlast angehangen bzw. überprüft.

(Quelle: VL Folien 9.37f und Mitschrift)

1b)“?~“, zugehöriger Bitstring: 0011111101111110 bzw. 11111101111110

**CRC16 berechnen**

Generatorpolynom: x16+x15+x2+1, entspricht 11000000000000101

Das Generatorpolynom ist von Grad 16, also hängen wir zunächst 16 0en an. Nun wird der Bitstring mit Anhang durch das Generatorpolynom dividiert (mit XOR). Der dabei entstehende Rest ist nun der Anhang also die Checksumme. Er entspricht 1000001100000100.

Bitstring mit Checksumme: 00111111011111101000001100000100

**Bitstuffing**

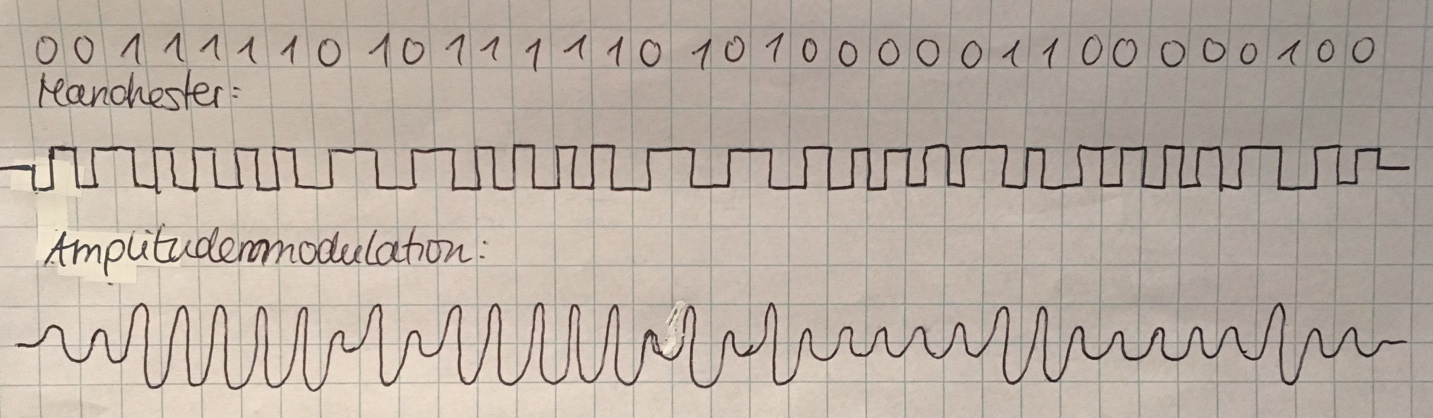
Die Flag ist 01111110, um sie zu unterschieden, wird bei der Nutzlast immer nach 5 Einsen eine 0 eingeführt. Unser String ändert sich also zu: 0011111010111110101000001100000100

**Manchester**

Bei der in der VL verwendeten Manchester Codierung handelt es sich um die nach G. E. Thomas. Dabei wird eine 0 durch einen Übergang von unten nach oben also 01 codiert und eine 1 als Übergang von oben nach unten also 10 codiert. Codiert lautet der String also: 01011010101010011001101010101001 100110010101010110100101010101100101 (unten nochmal als Zeichnung)

**Amplitudenmodulation**

1 ist hohe Amplitude, 0 ist niedrige (Frequenz ist egal):



2c)

Wie schwer ist es, dass zwei verschiedene Dateien gleicher Größe die gleiche Checksumme haben?

Eine Checksumme für c bits kann nur 2c einzigartige Werte annehmen. Die Anzahl der möglichen Nachrichten ist sehr viel größer, wodurch es zwangsweise zu Doppelungen kommt. Da die Checksummendoppelungen jedoch möglichst gut über die verschiedenen Datei/Nachrichtgrößen verteilt sind, ist es relativ schwer, dass zwei verschiedene Dateien gleicher Größe dieselbe Checksumme haben, wurden sie nicht wie hier extra dafür generiert.