Aufgabe 1: Data Link Layer

Wenn nicht anders angegeben, verwenden Sie pro Frage etwa 2–4 Sätze.

1. Nennen Sie mindestens drei Aufgaben des **Data Link Layers**.
2. Betrachten Sie den String bestehend aus den folgenden zwei Zeichen `~?` (Tilde, Fragezeichen).
   a) Geben Sie den Bitstring der Zeichen an. Nehmen Sie an, dass die Zeichen ASCII-kodiert sind.
   b) Ergänzen Sie den Bitstring um eine CRC-16-Prüfsumme. Geben Sie an, wie sie diese Prüfsumme berechnet haben.

      Den CRC den Sie benutzen sollen ist spezifiziert durch das Polynom `0x8005` bzw. $x^{16} + x^{15} + x^2 + 1$, jedoch sollen Sie davon ausgehen, dass die Eingabe LSB first ist. Zudem ist der Startwert des Shift-Registers `0xFFFF` und der finale xor-Wert auch `0xFFFF`.
   c) Wenden Sie **Bitstuffing** auf den (ergänzten) Bitstring an. Fügen Sie jedoch *keine* zusätzlichen **Flag Bits** am Anfang oder Ende hinzu.
   d) Zeichnen Sie den Spannungsverlauf auf, der entsteht, wenn der resultierende Bitstring mittels **Manchester-Kodierung** übertragen wird.

   Beachten Sie bei dieser Aufgabe die in den Folien verwendeten Definitionen für **Bitstuffing** und **Manchester-Kodierung**.

Aufgabe 2: TCP-Server

Sie haben im letzten Zettel schon Kommunikation zwischen zwei Prozessen mittels Unix-Domain-Sockets (als Stream-Sockets) implementiert. Jetzt sollen Sie das gleiche mittels TCP implementieren.

Hier erneut die gewünschte Funktionsweise des Servers und des Clients: Der Server soll in einer Endlosschleife Nachrichten von Clients empfangen, auf der Konsole ausgeben und dem Client eine Bestätigung (`OK`) zurückschicken. Der Client soll in einer Schleife Benutzereingaben einlesen, diese an den Server schicken und auf eine Antwort warten. Gibt der Benutzer das Wort `QUIT` ein, soll sich der Client beenden und eine eventuell existierende Verbindung sowohl auf Seite des Servers als auch des Clients schließen.

TCP Der Unterschied von Stream-Unix-Domain-Sockets zu TCP-Sockets besteht nur noch darin, dass statt an eine Datei an eine IP-Adresse gebunden (Server) bzw. verbunden (Client) wird. Weitere Informationen sind beispielsweise auch in der Manpage `man -s 7 tcp` zu finden. Implementieren Sie zwei Programme mit den Namen `tcp-server` und `tcp-client`. Der Aufruf der Programm erfolgt dann mittels:

```
$ ./tcp-server <address> <port>
$ ./tcp-client <address> <port>
```

Die Adresse beim Server sagt, auf welcher Adresse der Server lauscht. Bei Tests auf der gleichen Maschine ist `127.0.0.1` eine Möglichkeit. Für Server mit Portnummern kleiner als 1024 benötigt man Administratorrechte, verwenden Sie daher größere Portnummern.

TCP-Server erweitern Aktuell können Sie nur eine Verbindung gleichzeitig behandeln. Um das zu beheben gibt es zwei Möglichkeiten: Einerseits kann mittels `select` geschaut werden, auf welchem Socket zuerst neue Nachrichten ankommen. Andererseits kann der Server für jeden Client einen neuen Prozess mit `fork()` erstellen, der sich nur um die Abarbeitung des einen Clients kümmert, während der Elternprozess weiterhin nur auf neue Verbindungen wartet. Beim Empfangen von `QUIT` soll sich der Kindprozess beenden.