Bei uns war ursprünglich noch ein Dritter in unserer Abgabegruppe eingeteilt. Wir haben ihn vor über einer Woche versucht per E-Mail zu erreichen, leider erfolglos.
Nach Ablauf der Anmeldefrist zu den Abgabegruppen haben wir gesehen, dass diese Person leider unsere Abgabegruppe verlassen hat.
Bisher konnten wir noch keinen Dritten für unsere Abgabegruppe finden.

Aufgabe 1.1

i) **Verbindungsorientiert**. Sinnvoll, um beispielsweise die Authentifizierung (z.B. *Habe ich ein Abo?*) einmal durchzuführen. Damit verbraucht das eigentliche Streaming weniger Daten.
Unbestätigt. Hier braucht der Streamingserver keinerlei Bestätigung, dass die Streaming-Pakete erfolgreich beim Client eingetroffen sind.
Unzuverlässig. Da wir hier einen unbestätigten Dienst haben, können wir keinen zuverlässigen Dienst haben. Zusätzlich sind einzelne Übertragungsfehler oftmals ertragbar bei Streamingdiensten.

ii) **Verbindungsorientiert**. Sinnvoll, um zu überprüfen, ob das andere Konto überhaupt existiert.
Bestätigt. Der Bankkunde hat Interesse daran, dass die Überweisung auch ankommt.
Zuverlässig. Die Überweisung soll korrekt durchgehen und nicht verfälscht werden können.

iii) **Verbindungsorientiert**, falls die Beantragung der Briefwahl mit betrachtet wird. Hierbei muss der Bürger sich auch authentifizieren, bevor der eigentliche Datenaustausch (Briefwahl) stattfinden kann. Sonst **Verbindungslos**, da dies einen extra Schritt darstellen würde.
Bestätigt. Man möchte schon sicherstellen, dass seine Stimme gewertet wird. In der Praxis jedoch **Unbestätigt**.
Zuverlässig. Niemand soll die Wahlstimme verfälschen können (das Kreuz ändern). In der Praxis jedoch unmöglich.

Aufgabe 1.2

Ja, es kann keinen unbestätigten zuverlässigen Dienst geben. Ein zuverlässiger Dienst beruht auf Bestätigung, um die Korrektheit und Vollständigkeit der übertragenen Daten zu verifizieren. Ohne Bestätigung ist dies also nicht möglich.

Aufgabe 1.3

(1) `Con.Req; UnitData.Req(CR)`

(2) `UnitData.Ind(CC); Con.Cnf`

(3) `UnitData.Ind(DR); Dis.Ind`

(4) `UnitData.Ind(DR); Dis.Ind`

(5) `UnitData.Ind(CR); Con.Ind`

(6) `Con.Rsp; UnitData.Req(CC)`

(7) `Dis.Req; UnitData.Req(DR)`

(8) `Dis.Req; UnitData.Req(DR)`

Aufgabe 1.4

`g` ist eine **ICI**, da `g` nur eine Information ist, wie sich die Schicht des Alternating Bit Protocols zu verhalten hat. `g` wird also nicht horizontal kommuniziert.
Die Quittung `s` ist eine **PCI**, da diese nur für die Schicht des Alternating Bit Protocols interessant ist und auch horizontal kommuniziert wird.

Aufgabe 1.5

1. Sowohl auf Server als auch auf Client muss ein UDP IPv6 Socket erstellt werden, mit dem Daten empfangen / gesendet werden können:
   `int socket = socket(AF_INET6, SOCK_DGRAM, 0)`

2. Auf dem Server muss nun der socket zu einer Adresse `addr` gebinded werden:
   `bind(socket, addr, sizeof addr)`

3. Warten auf eine Nachricht vom Socket geht nun mit `recvfrom`.

4. Senden von Nachrichten vom Socket geht nun mit `sendto`.

5. Den Socket kann man nun schließen mit `close(socket)`.