Gesamtpunkte:

14/15

DatKom	Übungsblatt 1	Kaan Giray Buzluk	405099
SS 2021		Su Ada Yıldırım	410949
3. Mai 2021		Ozan Ege Şap	411851

Aufgabe 1.1 4/4,5

(i) Bei dem Streaming herunterladet man das Video ebenfalls nicht. Solange der Server die Video-Daten sendet, kann der Client es gleichzeitig lesen und das Video anschauen. Dabei braucht man einen schnellen Transport von Daten und es macht keine große Gefahr, wenn manche Pakets beim Transport verloren gehen. Also ist es unzuverlässig im Sinne der vollständigen Daten-Transport. Wir nehmen an, dass gleichzeitig 100 Personen einen Live-Konzert anschauen. Es macht Sinn für die Benutzer an jeder Zeit an den Live-Konzert zu verbinden. Daher braucht man keinen Kontext und ist es verbindungslos. Dabei hat der Server die Verantwortung nicht, dass die Daten erfolgreich transportiert sind. Der Server bekommt keinen Feedback von der Client zurück, ob er manche Pakets nicht bekommen hat. Damit ist es unbestätigt.

−0,5 Der Streaming Dienst benötigt aber Informationen z.B. über den zu verwendenden Codec, die Auflösung. Eine Verbindung ist also sehr sinnvoll.

(ii) Überweisung von Geld vom Konto eines Bankkundens auf das Konto eines anderen Bankkundens ist ein wichtiger Punkt, da es auf jeden Fall ohne Fehler passieren muss. Dabei hat die Zuverlässigkeit der Dienst eine hohe Wichtigkeit. Also muss es nicht sein, dass das überwiesenes Geld während dem Transport verloren geht. Dabei ist die Kommunikation zuverlässig sein. Bei einer Überweisung muss die Operation auf jeden Fall durchgeführt werden, also kann das Geld von einer wichtigen Sinne sein. Dabei soll es bestätigt sein um zu wissen, dass die Überweisung erfolgreich ist und keine weitere Wiederholungsprozesse nötig sind. Bei der Überweisung aber muss der Sender nicht auf eine Verbindung ewig warten, also kann er auf jeder Zeit die Überweisung senden. Also soll die Dienst verbindungslos sein.

(iii) Bei einer Briefwahl muss man der Inhalt des Briefes schützen und sicherstellen, dass keine ungültige Briefe vorkommen. Da es in hoher Priorität und Wichtigkeit ist, muss man den Prozess standardisieren. Es muss also zuverlässig sein, da jede Wahl einen Unterschied machen kann. Also jeder Brief muss auf jeden Fall zum Zentrum gehen und gezählt werden. Aber man bekommt keine Information darüber, ob der Brief erfolgreich transportiert ist. Deshalb ist es oft unbestätigt und da man auf keine Verbindung bei der Wahl wartet und einfach den Brief sendet, handelt es dabei von einer verbindungslosen Kommunikation.

Wäre aber hilfreich!

Aufgabe 1.2 1,5/2

Ob ein Dienst bestätigt oder unbestätigt ist sagt nichts über die Zuverlässigkeit der Datenübertragung aus. Eine Bestätigung dient nur dazu, dass der Sender erkennt, dass der Receiver seine Nachricht bekommen hat. Jedoch können Netzüberlastungen oder weitere Katostrophen wahr werden. In solchen Fällen, welche sehr oft auftritt, bringt eine Bestätigung nichts. Darüber muss der Dienst im Protokoll noch weitere Methoden implementieren, damit es sicher stellt, dass es die Daten zuverlässig übertragen hat. Zum Beispiel möchten wir einen Text-Datei zwischen zwei Rechnern übertragen. Erstmal teilen wir den Datei in gleichmäßigen Chunks. Also sind diese Chunks bestimmt geordnet. Dann

schicken wir zum anderen Rechner eine Information über die Anzahl der Chunks und vielleicht die Datengröße. Dann addiert der Sender am Ende der Chunks die Index-Nummer der jeweiligen Chunk. Also ist es in jedem Chunk ohne weiteren Kontext bestimmt, in welcher Stelle es kommt. Danach sendet er die Chunks. Nach der Übertragung überprüft der Receiver ob es genauso viele Chunks hat, wie es am Anfang in der Information gesagt wurde. Wenn ja, dann ist die Übertragung erfolgreich und shicken wir eine Bestätigung zurück. Falls es einige Chunks noch fehlen, dann weist man welche Chunks es braucht und schickt eine Bestätigung zurück, dass es noch eine Menge von bestimmten Chunks braucht. Damit wiederholt dieser Prozess bis alle Chunks erfolgreich übertragen wird. Damit handelt es von einer zuverlässigen Übertragung.

Nicht klar genug herausgestellt, ob die Bestätigungen nur Dienstintern genutzt werden oder nicht.
↳ Das differenziert dann den Bestätigten von unbestätigten Dienst.
-0,5

Aufgabe 1.3 4,5/4,5

Wir definieren:
CR := "Connection Request"
DR := "Disconnect Request"
CC := "Connection Confirmation"
PI := "Provider Abort Indication"
CRS := "Connection Response"

(1) Con.Req ; UnitData.Req(CR) ✓
(2) UnitData.Ind(CRS) ; Con.Cnf ✓
(3) UnitData.Ind(DR) ; Dis.Ind ✓
(4) UnitData.Ind(DR) ; Dis.Ind ✓
(5) UnitData.Ind(CR) ; Con.Ind ✓
(6) UnitData.Req(CRS) ; Con.Rsp (✓)
(7) UnitData.Ind(DR) ; Dis.Ind (✓)
(8) Dis.Req ; UnitData.Req(DR) ✓

Aufgabe 1.4 2/2

(i) Alternating Bit dient dazu zu bestimmen, welche Bestätigung zu welcher Nachricht gehört. Also werden die Nachrichten abwechselnd mit 0 und 1 markiert. Aber diese Praxis hängt von dem Protokoll ✓ ab. Damit steuert es den Protokollablauf und damit ein Protokollkontrollinformation (PCI). ✓

(ii) Die Verfälschungsbit ist hilfreich zum Erkennen von Übertragungsfehlern. Dabei ist es nutzbar zwischen Schichten ✓ und nur verantwortlich über die Integrität getragener Daten. Damit können Schichten unabhängig von dem Dateninhalt erkennen ob alles erfolgreich übertragen sind. Damit handelt es von einem Schnittstellenkontrollinformation (ICI). ✓

Aufgabe 1.5 2/2

Das UDP-Protokoll ist verbindungslos. Also jeder Instanz kann in jeder Zeit einen Datei senden oder zuhören. Dabei interagiert der Benutzer mit der Applikation und die Applikation dann nutzt das UDP-Protokoll, danach benutzt UDP IPV6 um Daten zu übertragen.

Der Prozess läuft über Sockets. Deshalb muss jeder Instanz, also Sender und Empfänger, einen Socket erzeugen. Dafür hat man den Syscall **socket(2)**.
Mit der Methode **int socket(int domain, int type, int protocol);** kann man den Socket bilden. Da wir UDP-Protokoll und IPV6 verwenden wollen, benutzen wir **AF_INET6** für domain und **SOCK_DGRAM** für type.

Also bildet man ein udp6 Socket mit
udp6_socket = socket(AF_INET6, SOCK_DGRAM, protocol);

Wenn ein Socket erzeugt wird, es existiert im name space und dem wird erstmal keine Adresse zugewiesen. Deshalb muss der Empfänger dann den Syscall **bind(2)** benutzen. Sonst kann der Empfänger kein Connection erhalten.

Nun sind die Instanzen bereit für die Übertragung. Der Sender benutzt den Syscall **sendto(2)** um die gewünschte Daten zu dem Empfänger zu senden. Datei benötigt man aber natürlich Informationen wie socket, message, length, flags, dest_addr, dest_len.

Der Empfänger soll nach der Adresse-Bindung die Daten zuhören. Bei einer solchen verbindungslosen Kommunikation benutzt man den Syscall **recvfrom(2)**. Wenn es ein Paket kommt, ist der Inhalt des Pakets in einem Buffer zugefügt. Also blockiert der Prozess bis das Datagram erhalten ist. Danach bleibt nichts zu tun. Man kann danach mit dem Syscall **close(2)** die Sockets schließen.