Question 1.

a) After some initial negotiation, A creates a data packet, saves a copy of it, and sends the data to B, and also sets a timer; B then replies with an acknowledgement (ACK). Sender A always waits for each ACK before sending the next data packet. ACKs that do not reach A before the timer expires imply that the data packet is presumed lost, and must therefore be retransmitted (therefore the copy of the data packet above).

b) If the communication channel is unreliable and loses or delays some packets, there could be duplicate packets arriving at the receiver, so the receiver needs to be able to distinguish between a new and a duplicate of an old packet.

c) - service = what a lower layer provides to an upper layer, for example “send” and “receive”

- protocol = the implementation of one “box” in the OSI reference model, and also the rules for the communication between peers (two “boxes” at the same level in the network)

Question 2.

a) “Due to its high overheads TCP is not suitable for client-server applications.”

- Both False and True. It depends on the application. It is true that TCP has a high overhead, but this overhead provides the application with reliable data transfer, and depending on the client-server application this is good or bad. VoIP (see below) for example would not use TCP.

b) “Internet telephony typically uses UDP as the underlying transport level protocol because UDP can provide the end-to-end delay guarantee“.

- False. Even though VoIP typically does use UDP this does not provide the application with an end-to-end guarantee; this functionality must be implemented in the application itself. However, UDP does provide the application with a smaller delay.

Question 3.

a) The IP addresses that were handed out to organizations et al. where previously divided into classes A, B, and C. For every network of a campus for example, the class C address is at least needed. If a network has more than 255 nodes, it needs a class B address. There is a finite number of network addresses in the class system, and even fewer class B addresses than class C addresses. Class B addresses were popular because they could provide addressing for more than 255 networks and “you never know when you are going to hit the maximum of 255 nodes of a class C address”. Therefore, the space of class B addresses was being exhausted. The class system suffered from an address assignment inefficiency.
The problem was solved by disregarding from the class system and instead handing out chunks of addresses, together with a netmask and using subnetting.

b) packet fragmentation is the process when a bigger packet is divided into smaller packets (fragments). It is used because different networks have different MTUs (maximum transfer units), and needed because a host normally does not know the maximum allowed packet size for all the different networks involved in the communication.

**Question 4.**

a) with circuit-switching we get: the path through the network is established during the connection setup, which takes 100 msec; after that A can transmit with 2 Mbps directly to D without delay (delay negligible as stated in the assignment); this gives us 1 Mbit / 2 Mbps = 500 msec for transmitting the message, for a total of 500 + 100 = 600 msec.

b) with packet-switching we get: first, assume B and C are store-and-forward switches. (you can also assume variants of cut-through switches; no-delay switches are a bit futuristic, but ok …)

The 1 Mbit message is divided into packets, that is, 1Mbit / (250 x 8) = 500 packets; each packet needs a header of 50 bytes, which means that one packet is 300 bytes and can be transmitted A -> B in (300 x 8) bits / 2 Mbps = 1.2 msec. This is the delay of a switch in our assumption (a switch needs to wait for the whole packet before retransmitting it). Every 1.2 msec A can output a packet to the link without bottlenecks, which gives us 500 packets à 1.2 msec = 600 msec for the whole message as packets with headers. At the receiving side at D we need to wait for the last 3 packets to arrive (which are still out there on the links); this will take another 3 x 1.2 = 3.6 msec, which gives us a total transmission time of 600 + 3.6 = 603.6 msec.

**Question 5.**

a) The slow start algorithm is used mainly when a new connection is opened. It means that the TCP sender begins by transmitting at a slow rate but increases its sending rate exponentially fast. Initially, the Congestion Window is set to one MSS. Every RTT the Congestion Window is increased by doubling the value, until there is a loss event, at which time the Congestion Window is cut in half (and then grows linearly).

b) Fast retransmission is triggered when three duplicate ACKs arrive at the sender. This heuristic is used in order to avoid going back to doing slow start (see a) above). It means that instead of going back to 1 MSS per RTT (slow start), the Congestion Window is cut in half and then increases linearly.

**Question 6.**

a)

We get count-to-infinity, with infinity defined as 8.

Assumption: The numbers are each router’s distance to D and link costs are 1.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>1 initially</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>
7 exchanges are required.

b)
Any topology with a *loop*; split horizon does not work in networks that contain loops, such as this:

```
       D
      / \
     /   \
    /     \
   /       \
  A-------B-------C
      |         / \
      |        /   \
      |       /     \
      |      /       \
      |     /         \
      |    /           \
      |   /             \
      |  /               \
      | /                 \
      |/                   \
     E
```

**Question 7.**

a) In order to solve the problem of scalability, a distributed system needs to address three aspects:

i) *hide communication latencies*: DNS queries are asynchronous, using UDP, which means that after requesting a service to resolve a domain name, the client can do other useful work.

ii) *distribution*: DNS is divided into zones, each of which is managed by a couple of name servers (one is backup); when a DNS query is issued, each part of a domain name from right to left determines to which zone the query should be forwarded.

iii) *replication*: DNS makes use of caching, a special case of replication, in the local name server in order to avoid propagating a DNS query too far up in the name server hierarchy.

b) An example of horizontal distribution is a web server replicated across several machines in a LAN. Each server has the same set of web pages, and each time a web page is updated, a copy is immediately places at each server. When a request comes in, it is forwarded to a server using a round-robin policy.

**Question 8.**

a) One example of a CORBA service is the *naming* service, by which objects can be given a human-readable name that maps to the object’s identifier.

b) Objects and services in CORBA are specified using IDL, the Interface Definition Language.