Problem 1 RPC Failure Semantics

- **Client** The client has to give each message a unique ID by which it can be identified. The client has to hold a buffer containing the message until it (the client) received answer from the server. The client resends the message every *timeout* seconds. As soon as the client receives the answer from the server, he can dispose the message stored. The client also has to perform Marshalling on the message if necessary.
- **Server** The server has to hold a buffer containing the IDs of all received messages. If the server receives a message with an unknown ID, the ID is added to the buffer and the message processed. If the server receives a message with an ID he already has in its buffer, he simply disposes the message. The server also has to perform Demarshalling on the message if necessary. The server also has to send a (marshalled) reply to the client if requested.

While this is sufficient to ensure *At-Most-Once*, one should note that the server should forget every ID after some time. Otherwise, due to the amount of IDs being limited, he will receive a message with an ID he has already seen, but with different content. This would be disposed, even though it should be processed. This would however still satisfy *At-Most-Once*, as the message is 0 times (which is <=1) processed.

Problem 2 Marshalling

- **Definition** Marshalling describes the process of converting complex data (for example objects containing objects or objects in general) into a more simple data format like a byte-sequence. Often, knowledge is lost in the presentation and needs to be supplied from the outside. In the CORBA CDR, all type information and order is lost, a receiving process needs to know it already to reassemble the represented complex data. This process is also required to unify the data reprasantation between different operatongs systems, programming languages or hardware architectures.
- **CORBA** CDR is often implemented on top of the RMI protocol. The target and the invoker usually know what type to expect in the messages carrying the arguments and the results. In this case, additional typing would only result in overhead of unnecessary space and time cost. However, if the types are unknown to the target or the invoker, this will be a problem because the information could (at least partly) be undecipherable. Thus, CORBAs advantages are small speed and space requirements during transmission, but knowledge about the data already be present on the receiver-side.

Problem 3 Request-Reply Protocol

- **Advantage** The server can delete information about the clients request once the ACK reaches the server.
- **Disadvantage** ACK causes additional traffic and could also get lost, in which case the server will never be notified that the ACK arrived. If RRA is used the server should also have a timeout for the possible ACK's. To prevent overflowing of the servers cache, if the server receives no ACK from the client for some time (timeout), he can simply clear its cache.

The server should not send the reply multiple times when not receiving an ACK. If the reply gets lost, the client will simply resend its request, for which the server can send a cached result again, as he already computed the request before. It is usually cheaper to send multiple requests over the network than a possibly big result.