Assignment 2

Georgia Koloniari
Salteas-Kalogeras Panagiotis

Problem 3:
Since the service requires reliable communication it should not be part of the lower-layers because they do not offer such reliability. In contrast, the transport layer does offer reliability that a middleware service can use. Middleware protocols support high-level communication services and so this service naturally belongs to this layer. We could try to place it also at the transport layer which offers reliability, but scalability can be guaranteed only if the specific application requirements are taken into account.

Problem 6:
This approach can work just fine without causing any confusion or problems. We just have to clarify that there is never any confusion as to which is the first byte of a message (packet): the first byte sent is always the first byte received. The confusion arises when we try to discern the first byte of a word (collection of bytes) and not of a message. For example, in four bytes representing a 32-bit integer, which is the byte that should be considered the most significant byte. The only thing that remains for the sender and the receiver to do is not try to interpret any parts of the message containing the first byte as a multi-byte integer. After the code has been read the receiver can transform the data to its machine's native form, and the representation issue is solved.

Problem 8:
The main drawback of this approach is that it has no flexibility. A change in the endpoint on a server machine would mean that the program must be rewritten. We no longer have the flexibility of the daemon that would assign each server dynamically a free port. This problem would become more intense if we had many server programs running on the same machine. If we also pass the directory server then if the server changes its network address the program needs to be changed to reflect the new server address again. Obviously it is very difficult to dynamically allocate objects to servers that are always assigned to the same endpoint and thus this approach should not be used.

Problem 9:
An example of an object reference that allows a client to bind to a transient remote object is implemented by CORBA systems.

Problem 15 :
Consider a synchronous send primitive. A simple implementation is to send a message to the server using asynchronous communication, and afterwards let the sender continuously poll for an incoming acknowledgement or response from the server. If the local operating system stores incoming messages into a local buffer, then an alternative is to block the sender until it receives a signal from the operating system that a message has arrived, after which it can execute an asynchronous receive.

Problem 22:
Transient synchronous communication is inappropriate especially in terms of geographical scalability. Since the distance between the sender and the receiver increases and network delays become bigger the blocking time increases unacceptably. A reasonable timeout cannot be defined since network delays cannot be guaranteed and we may cause the same message to be send multiple times although the receiver eventually gets the initial message, thus increasing the network traffic without a reason. The problem can be solved if we use non-blocking send and receive primitives and thus asynchronous communication. We can also use weaker forms of transient synchronous communication such as asynchronous RPCs or deferred synchronous operations.

Problem 24:
To be able to guarantee a maximum end-to-end delay in the ring we will use a token. When the token circulates the ring, each computer is permitted to send data across the ring (in the same direction as the token) only when it is holding the token. If we limit the time a computer can hold the token, so that no computer will be allowed to hold the token for more than *T* seconds and we suppose that communication between two adjacent computers is bounded to T_1, then the token will have a maximum circulation time, which corresponds to a maximum end-to-end delay for each packet sent. For a collection of *n* computers the maximum delay would be $n * (T + T_1)$.

Problem 26:
Let *t* be the length of the maximum burst interval. In the extreme case, the bucket is full at the start of the interval so it contains 1 million bytes (1 Mbyte) and another (*token bucket rate*) * (*maximum burst interval*) = 10 * *t* Mbytes come in during the interval. The output during the transmission burst contains (*maximum transmission rate*)* (*maximum burst interval*) = 50 * *t* Mbytes. From the above two equations we have: 1 + 10 * *t* = 50 * *t*. $\Rightarrow$ *t* = 25 msec.