Q1: Concurrency transparency is a desirable goal for distributed systems. Can we assume that such a property would be automatically provided by centralized distributed systems? Discuss.

Q2: In many communication systems, calls to send set a timer to guard against hanging the client forever if the server crashes. Suppose that a fault-tolerant system is implemented using many processors for all clients and servers. As a result, the probability for a client or a server to crash becomes effectively zero. Do you think it is safe to fully eliminate timeouts in such a system?

Q3: How do mobile agents compare to static clients making remote invocations? Can web crawlers be deemed as mobile agents? Discuss.

Q4: As a distributed systems developer, what would you adopt for multimedia data streams, a synchronous or asynchronous distributed system? Justify your answer.

Q5: Encryption and authentication are used to build secure channels as a service layer on top of existing communication services. In addition, designers of distributed systems encapsulate messages with timestamps for transmission over secure channels. Discuss why would this latter step be required?

Q6: Explain how could a transmitted sequence of packets through a wide area network arrive unordered at their destination (i.e., in an order that differs from that in which they were sent). Why can’t this happen in a local area network?
Q7: It is claimed that one way to handle parameter conversion in RPC systems is to have each sender machine sending parameters in its native representation, with the receiving machine doing the translation (if necessary). Can this work? Discuss.

Q8: A client makes remote procedure calls to a server. The client takes 5 milliseconds to compute the arguments for each request, and the server takes 10 milliseconds to process each request. The local operating system processing time for each send or receive operation is 0.5 milliseconds, and the network time to transmit each request or reply message is 3 milliseconds. Marshalling or unmarshalling takes 0.5 milliseconds per message.

Calculate the time taken by the client to generate and return from two requests:

(i) if it is single-threaded, and
(ii) if it has two threads that can make requests concurrently on a single processor.

You can ignore context-switching times. Is there a need for asynchronous RPC if client and server processes are threaded?

Q9: Devise a scenario in which multicasts sent by different clients are delivered in different orders at two group members. Assume that some form of message retransmissions is in use, but that messages that are not dropped arrive ordered. Suggest how recipients might remedy this situation.

Q10: Message passing is both time- and space-coupled – that is, messages are both directed towards a particular entity and require the receiver to be present at the time the message was sent. Consider the case where messages are directed towards a name rather than an address. Assume that this name is resolved using DNS.

(i) Does such a system exhibit the same level of indirection?
(ii) State the main difference between message passing and message queuing.
Q11: If a communication paradigm is asynchronous, is it also time-uncoupled? Explain your answer with examples as appropriate.

Q12: Use the UDP program you developed in Recitation 1 to make:

(i) A test kit to determine the conditions in which datagrams are sometimes dropped.
(ii) A client program that repeatedly reads a line of input from the user, sends it to the server in a UDP datagram message, then receives a message from the server. The client sets a timeout on its socket so that it can inform the user when the server does not reply.