Where an algorithm is asked for, you may write in any suitable pseudocode. Correct syntax for any computer language is not expected.

Answer 3 questions.

1.

a) Ideally, we would like to ensure that

- messages are delivered in the order sent
- every message sent is delivered exactly once
- an acknowledgment is received for every message sent

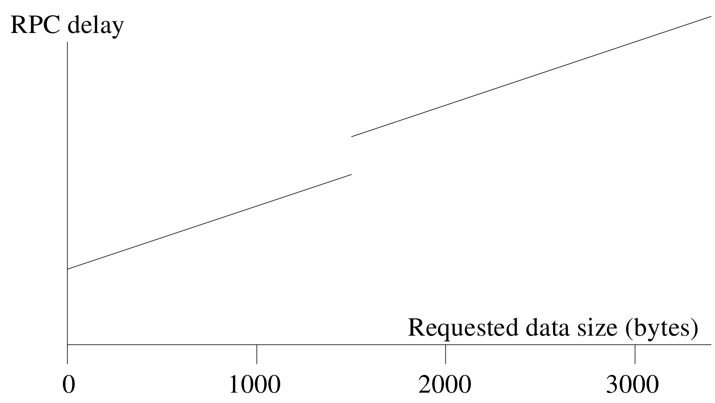
i) In the presence of crashes and unreliable message transmission, it is expensive to achieve at least the last two of these. Why? [6]

ii) If we wish to have cheap operation it may be necessary to choose between delivering messages more than once or failing to deliver some. Which of these courses would, in general, be preferable and why? Give examples both of situations which support your argument and those where it will fail. [10]

b) In fewer than 150 words, outline the steps in transferring a structured RPC argument across a network to a remote procedure. [6]

c) What factors contribute to the end-to-end delay incurred in sending an RPC besides network transmission times? [6]

d) In the light of your answer to part c), interpret the following graph, paying particular attention to the reason for the discontinuity:



[5]

[TURN OVER]

2.

a) What are the conditions that must be met if termination is to be declared in a distributed program? [4]

b) Present pseudocode and a correctness argument for an approach to termination detection based on token passing. Be explicit about your assumptions. [9]

c) In the case where message transmission across channels is asynchronous and unordered, it is still possible to detect termination using tokens, provided that one has more information, for example, if messages are acknowledged. Assuming that message delivery is reliable, but not necessarily sequenced, devise a termination detection algorithm which uses acknowledgments. In doing this, you should:

i) Say what messages are transferred between nodes.

ii) Say what the actions of a node are on receipt of the above message types

iii) Give an informal argument for the correctness of your approach. [20]

3.

a) Explain what you understand by the terms *Secret Key Cryptography* and *Public Key Cryptography*. [10]

b) Explain why *Public key Certificates* are needed and identify their main components. [5]

c) You are asked to devise a method of two-way authentication between users and computer systems to replace the current one-way authentication provided by logging-in using a password. It is to be based on each user having a smart-card which can be down-loaded with the public key certificates of those systems they need to use, and using a challenge-response protocol.

i) Identify the likely threats in such a system [3]

ii) Devise a suitable protocol and state it using the Needham and Schroeder notation [10]

iii) Explain how the protocol provides the needed services and counters the threats [5]

[CONTINUED]

4.

a) Define what are meant by the ACID properties of a transaction. [4]

b) The UniCol company wishes to extend its distributed file system to support transactional operation. In order to do this, it wishes to provide several operations:

- Start_transaction() → TID
- Commit_transaction(TID)
- Abort_transaction(TID)
- TOpen(TID, filename) → file_descriptor
- TClose(TID, file_descriptor)
- TWrite(TID, file_descriptor, data)
- TRead(TID, file_descriptor, length) → data
- TDelete(TID, file_descriptor)
- TLength(TID, file_descriptor) → length

In all of the above operations, any changes to files are made in a tentative way until the transaction commits.

Describe how you would implement such operations, taking into account the fact that the system should be scaleable. If there are any operations which you need to assume that the underlying file system already has, say what they are. [20]

c) Show that your implementation of the operations described above satisfies the ACID properties, or else argue that it would be undesirable for them to do so. [9]

[TURN OVER]

5. In a distributed system supporting a hierarchically arranged partitioned namespace, the resolution of a name (e.g. /a/b/c/d) involves searching various pathname components one-by-one (i.e. / then a, then b, etc.) in the nameservers concerned. It is not unreasonable to expect that details of each of these components are held on different machines. As a result, successful resolution requires that all of these nodes are available and accessible by the client. Thus reliability is very important in name resolution in distributed systems.

   Equally, however, not all objects are of equal importance to all the users of a distributed system at all points in time. Thus, a high degree of reliability for all elements of the naming system over all time would be unduly expensive, and is not required. Likewise, an approach which requires absolute consistency between replicas of nameservers at all times is considered too expensive. Relaxing this requirement would allow communication overheads to be reduced or deferred, at the cost of decreased reliability on those entries affected whilst copies remain inconsistent.

   Consequently, what is required is a name resolution mechanism in which the degree of reliability is tuneable for different objects, but also in which the communication costs and the degree of consistency necessary for a given object can be taken into consideration in deciding when to propagate any given update or where to place replicas. It must be possible dynamically to change the degree of reliability, the degree of consistency and the placement of replicas as usage and prevailing conditions change.

   Propose and justify a naming and location mechanism with the above properties, taking care to identify its advantages and disadvantages as compared to alternative approaches. In answering this you should provide examples to illustrate any major points you wish to make. Would you recommend your system for use in all circumstances? [33]

[END OF PAPER]